

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE GENCPM
 OBJECT MODULE PLACED IN GENCPM.OBJ
 COMPILER INVOKED BY: PLM80 GENCPM.PLN DEBUG OPTIMIZE

```

1      $title('CP/M 3 System Generation')
      gencpm:
      do;

/*
      Copyright (C) 1982
      Digital Research
      P.O. Box 579
      Pacific Grove, CA 93950
*/

/*
      Revised:
      02 Dec 82 by Bruce Skidmore
*/

2 1      declare true literally 'OFFH';
3 1      declare false literally '0';
4 1      declare forever literally 'while true';
5 1      declare boolean literally 'byte';
6 1      declare cr literally '0dh';
7 1      declare lf literally '0ah';

8 1      mon1:
          procedure (func,info) external;
9 2          declare func byte;
10 2         declare info address;
11 2         end mon1;

12 1      mon2:
          procedure (func,info) byte external;
13 2          declare func byte;
14 2          declare info address;
15 2          end mon2;

16 1      relfix:
          procedure byte external;
17 2          end relfix;

18 1      setbuf:
          procedure external;
19 2          end setbuf;

20 1      getdef:
          procedure external;
21 2          end getdef;

22 1      crtdef:
          procedure external;
23 2          end crtdef;

```

CP/M -PLUS V3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # CP3-135

```

24 1      declare reset label external;

25 1      declare fcb (1) byte external;
26 1      declare fcb16 (1) byte external;
27 1      declare tbuff (1) byte external;
28 1      declare maxb address external;
29 1      declare bitmap (128) byte external;

30 1      declare FCBin address public;

31 1      declare bios$fcbl (36) byte initial (
           0,'BNKBIO$3','SPR',0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

32 1      declare res$fcbl (36) byte initial (
           0,'RESBIO$3','SPR',0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

33 1      declare bnk$fcbl (36) byte initial (
           0,'BNKBIO$3','SPR',0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

34 1      declare FCBout (36) byte initial (
           0,'CPM3    ','SYS',0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

35 1      declare data$fcbl (36) byte public initial (
           0,'GENCPM  ','DAT',0,0,0,0,0,0,0,0,
           0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

36 1      declare offset byte public;
37 1      declare prgsiz address public;
38 1      declare bufsiz address public;
39 1      declare codsiz address public;
40 1      declare bios$pg byte public;
41 1      declare sch$pg byte public;
42 1      declare res$pg byte public;
43 1      declare bnk$pg byte public;
44 1      declare bnk$off byte public;
45 1      declare res$len byte public;
46 1      declare non$bnk byte public;
47 1      declare dma address public;

48 1      declare hexASCII (16) byte public data (
           '0123456789ABCDEF');

49 1      declare lnbfr (14) byte public initial (12);

50 1      declare scbfr (1) structure (
           record (128) byte) public at (.memory);

51 1      declare fcb$msg (13) byte initial ('      . $');

52 1      declare query boolean public;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

/*

B D O S Procedure & Function Calls

*/

```
53 1  system$reset;
      procedure public;
54 2      call mon1 (0,0);
55 2  end system$reset;

56 1  write$console:
      procedure (char) public;
57 2      declare char byte;
58 2      if display then
59 2          call mon1 (2,char);
60 2  end write$console;

61 1  print$console$buffer:
      procedure (buffer$address) public;
62 2      declare buffer$address address;
63 2      if display then
64 2          call mon1 (9,buffer$address);
65 2  end print$console$buffer;

66 1  read$console$buffer:
      procedure (buffer$address) public;
67 2      declare buffer$address address;
68 2      declare buf based buffer$address (1) byte;
69 2      buf(1) = 0;
70 2      if automatic then
71 2          do;
72 3          if not query then
73 3              return;
74 3          end;
75 2      call mon1 (10,buffer$address);
76 2      buf(buf(1)+2) = 0;
77 2  end read$console$buffer;

78 1  crlf:
      procedure public;
79 2      call write$console (cr);
80 2      call write$console (lf);
81 2  end crlf;

82 1  error:
      procedure(terr$code,err$type,err$msg$adr) public;
83 2      declare (terr$code,err$type) byte;
84 2      declare err$msg$adr address;
85 2      display = true;
86 2      call print$console$buffer (.(cr,lf,
      'ERROR: $'));
87 2      call print$console$buffer (err$msg$adr);
88 2      if err$type = 1 then
89 2          call print$console$buffer(.(fcb$msg);
90 2      call crlf;
91 2      if terr$code then
92 2          call system$reset;
93 2      if automatic and not query then
94 2          do;
95 3          fcb(1),
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

        fcb16(1) = ' ';
96 3      goto reset;
97 3      end;
98 2      end error;

99 1      open$file:
        procedure (fcb$address) byte public;
100 2      declare fcb$address address;
101 2      declare fcb based fcb$address (1) byte;
102 2      fc_12), /* ex = 0 */
        fcb(32) = 0; /* cr = 0 */
103 2      return mon2 (15,fcb$address);
104 2      end open$file;

105 1      close$file:
        procedure (fcb$address) public;
106 2      declare fcb$address address;
107 2      call mon1 (16,fcb$address);
108 2      end close$file;

109 1      delete$file:
        procedure (fcb$address) public;
110 2      declare fcb$address address;
111 2      call mon1 (19,fcb$address);
112 2      end delete$file;

113 1      read$record:
        procedure (fcb$address) public;
114 2      declare fcb$address address;
115 2      if mon2 (20,fcb$address) <> 0 then
116 2      do;
117 3      call error(true,1,('
        'Reading file: '$'));
118 3      end;
119 2      end read$record;

120 1      write$record:
        procedure (fcb$address) public;
121 2      declare fcb$address address;
122 2      if mon2 (21,fcb$address) <> 0 then
123 2      do;
124 3      call error(true,1,('
        'Writing file: ', '$'));
125 3      end;
126 2      end write$record;

127 1      create$file:
        procedure (fcb$address) public;
128 2      declare fcb$address address;
129 2      declare fcb based fcb$address (1) byte;
130 2      if mon2 (22,fcb$address) = 255 then
131 2      do;
132 3      call error(true,0,('
        'Directory full', '$'));
133 3      end;
134 2      fcb(32) = 0; /* set cr = 0 */
135 2      end create$file;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

136 1      set$DMA$address:
        procedure (DMA$address) public;
137 2          declare DMA$address address;
138 2          call mon1 (26,DMA$address);
139 2      end set$DMA$address;

140 1      read$random$record:
        procedure (fcb$address) public;
141 2          declare fcb$address address;
142 2          if mon2 (33,fcb$address) <> 0 then
143 2              do;
144 3                  call error(true,1,.(
                    'Reading file: ','$'));
145 3              end;
146 2      end read$random$record;

147 1      write$random$record:
        procedure (fcb$address) public;
148 2          declare fcb$address address;
149 2          if mon2 (34,fcb$address) <> 0 then
150 2              do;
151 3                  call error(true,1,.(
                    'Writing file: ','$'));
152 3              end;
153 2      end write$random$record;

154 1      set$random$record:
        procedure (fcb$address) public;
155 2          declare fcb$address address;
156 2          call mon1 (36,fcb$address);
157 2      end set$random$record;

/*
  Data Structures
*/

158 1      declare automatic boolean;
159 1      declare display boolean public;

160 1      declare nmb$sect address;

161 1      declare link address at (.memory);

162 1      declare bios$atts(3) address public;
163 1      declare res$atts(3) address public;
164 1      declare bnk$atts(3) address public;

165 1      declare res$bios$len byte public;
166 1      declare res$base byte public;
167 1      declare pg$dif byte public;
168 1      declare xmove$implemented boolean public;
169 1      declare os$top address;

170 1      declare system$data (256) byte;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

171 1 declare common$len byte public at (.system$data(1));
172 1 declare banked$len byte public at (.system$data(3));
173 1 declare sys$entry address public at (.system$data(4));

174 1 declare prt$msg$ptr byte;

175 1 declare dont$hash boolean;

176 1 declare wordadr address;
177 1 declare word based wordadr address;

178 1 declare len byte;
179 1 declare off address;
180 1 declare res$flg byte;
181 1 declare save$mem$top byte;

182 1 declare drvtbl$adr address public;
183 1 declare drvtbl based drvtbl$adr (16) address;

184 1 declare dph$adr address public;
185 1 declare dph based dph$adr structure (
    xlt      address,
    scratch1(4) address,
    scratch2  byte,
    mf        byte,
    dpb       address,
    csv       address,
    alv       address,
    dirbcb    address,
    dtabcb    address,
    hash      address,
    hbank     byte);

186 1 declare dpb$adr address public;
187 1 declare dpb based dpb$adr structure (
    spt      address,
    bsh      byte,
    blm      byte,
    exa      byte,
    dsa      address,
    dra      address,
    al0      byte,
    al1      byte,
    cks      address,
    off      address,
    psh      byte,
    pha      byte);

188 1 declare bnk$swt boolean external;
189 1 declare dbl$alv boolean external;
190 1 declare mem$top byte external;
191 1 declare bnk$top byte external;
192 1 declare lerror boolean external;
193 1 declare bdrive byte external;
194 1 declare con$wid byte external;
195 1 declare con$pag byte external;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

196 1 declare bck$spc boolean external;
197 1 declare rubout boolean external;
198 1 declare prt$msq boolean external;
199 1 declare hash(16) boolean external;
200 1 declare num$seg byte external;
201 1 declare crdatf boolean external;

```

```

202 1 declare mem$tbl (17) structure(
      base      byte,
      len       byte,
      bank      byte,
      attr      address) external;

```

```

203 1 declare record(16) structure(
      size      address,
      attr      byte,
      altbnks   byte,
      no$dirrecs byte,
      no$dtaresc byte,
      ovlydir$dr byte,
      ovlydta$dr byte,
      dir$resp  byte,
      dta$resp  byte) external;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

204 1 declare quest(157) boolean external;

205 1 declare hash$data(16) address public;
206 1 declare hash$space address public;
207 1 declare alloc(16) address public;
208 1 declare alloc$space address public;
209 1 declare chk(16) address public;
210 1 declare chk$space address public;

```

```

/*
   Local Procedures
*/

```

```

211 1 movef:
      procedure (count,source$adr,dest$adr) public;
212 2   declare count byte;
213 2   declare (source$adr,dest$adr) address;

214 2   if count = 0
      then return;
216 2   else call move (count,source$adr,dest$adr);

217 2   end movef;

218 1 shift$left:
      procedure (pattern, count) address public;
219 2   declare count byte;
220 2   declare pattern address;

221 2   if count = 0
      then return pattern;
223 2   else return shl(pattern,count);

```

```
224 2      end shift$left;

225 1      upper;
226 2          procedure(b) byte public;
227 2              declare b byte;

228 2              if b < ' ' then return cr; /* all non-graphics */
229 2              /* translate alpha to upper case */
230 2              if b >= 'a' and b <= 'z' then
231 2                  b = b and 101$1111b; /* upper case */
232 2              return b;
233 2      end upper;

234 1      valid$drive;
235 2          procedure(drv) boolean public;
236 2              declare drv byte;
237 2              if (drv >= 0) and (drv <= 15) then
238 2                  return true;
239 2              call error(false,0,('Invalid drive.$'));
240 2              return false;
241 2      end valid$drive;

242 1      get$response;
243 2          procedure (val$adr) public;
244 2              declare val$adr address;
245 2              declare val based val$adr byte;
246 2              call write$console (' ');
247 2              if val = 0ffh
248 2                  then call write$console ('Y');
249 2                  else call write$console ('N');
250 2              call print$console$buffer (('.' ? ',','$'));
251 2              call read$console$buffer (,lnbfr);
252 2              if lnbfr(1) = 0
253 2                  then return; /* accept default */
254 2              val = (upper(lnbfr(2)) = 'Y');
255 2      end get$response;

256 1      dsply$hex;
257 2          procedure (val) public;
258 2              declare val byte;
259 2              call write$console (hexASCII(shr (val,4)));
260 2              call write$console (hexASCII(val and 0fh));
261 2      end dsply$hex;

262 1      dsply$hex$adr;
263 2          procedure (val) public;
264 2              declare val address;
265 2              call write$console (' ');
266 2              call dsply$hex (high (val));
267 2              call dsply$hex (low (val));
268 2              call write$console ('H');
269 2      end dsply$hex$adr;

270 1      dsply$hex$high$adr;
271 2          procedure (val) public;
272 2              declare val byte;
273 2              call dsply$hex$adr (double (val)*256);
```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

268 2      end dsply$hex$high$adr;

269 1      dsply$dec$adr:
          procedure (val) public;
270 2      declare val address;
271 2      declare big address;
272 2      declare (digit,i) byte;
273 2      declare pdigit boolean;

274 2      pdigit = false;
275 2      digit = '0';
276 2      big = 10000;
277 2      if val = 0 then
278 2          call write$console(digit);
          else
279 2          do;
280 3              do i = 0 to 4;
281 4                  do while val >= big;
282 5                      pdigit = true;
283 5                      digit = digit + 1;
284 5                      val = val - big;
285 5                      end;
286 4                      if pdigit then
287 4                          do;
288 5                              call write$console(digit);
289 5                              digit = '0';
290 5                              end;
291 4                              big = big / 10;
292 4                          end;
293 3                      end;
294 2      end dsply$dec$adr;

295 1      dsply$param:
          procedure (val,base) public;
296 2      declare (val,base) byte;
297 2      call write$console (('');
298 2      if base = 10 then
299 2          do;
300 3              call write$console ('#');
301 3              call dsply$dec$adr(double(val));
302 3          end;
          else
303 2          do;
304 3              call dsply$hex (val);
305 3          end;
306 2      call print$console$buffer (('') ? ',','$');
307 2      end dsply$param;

308 1      get$param:
          procedure (string$adr,val$adr,pbase) public;
309 2      declare (string$adr,val$adr) address;
310 2      declare pbase byte;
311 2      declare base byte;
312 2      declare val based val$adr byte;
313 2      declare string based string$adr (1) byte;
314 2      declare char byte;
315 2      declare lbindx byte;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

316 2      prompt$read;
           procedure;
317 3          call print$console$buffer (string$adr);
318 3          if string(0) = ' ' then
319 3              do;
320 4              call write$console ((' ');
321 4              call dsply$hex (val);
322 4              do lbindx = 1 to 2;
323 5                  val$adr = val$adr + 1;
324 5                  if (lbindx=2) and (not bnk$swt) then
325 5                      do;
326 6                          val = 0;
327 6                      end;
                       else
328 5                          do;
329 6                              call write$console (',');
330 6                              call dsply$hex (val);
331 6                          end;
332 5                      end;
333 4                      val$adr = val$adr - 2;
334 4                      call print$console$buffer (.(') ? ', ' $'));
335 4                  end;
                       else
336 3                      do;
337 4                          call dsply$param (val,pbase);
338 4                      end;
339 3                      base = 16;
340 3                      lbindx = 1;
341 3                      call read$console$buffer (,lnbfr);
342 3                  end prompt$read;

343 2      call prompt$read;
344 2      if lnbfr(1) = 0 then
345 2          do;
346 3              /* accept default value */
347 3              call crlf;
348 3              return;
349 2          end;
350 2          val = 0;
351 2          do while (char := upper(lnbfr(lbindx:=lbindx+1))) <> cr;
352 3              if char = ',' then
353 4                  do;
354 5                      val$adr = val$adr + 1;
355 5                      val = 0;
356 5                      base = 16;
357 4                  end;
358 3              else
359 4                  do;
360 5                      if char = '#' then
361 6                          base = 10;
362 5                      end;
363 4                  else
364 5                      do;
365 6                          char = char - '0';
366 6                          if (base = 16) and (char > 9) then

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SFR. #

```

365 5      do;
366 6          if char > 16
              then char = char - 7;
              else char = 255;
368 6      end;
369 6      if char < base then
370 5      do;
371 5          val = val*base + char;
372 6      end;
373 6      else
374 5      do;
375 6          char,
          val = 0;
376 6          call error (false,0,.(
              'Bad character, re-enter $'));
377 6          call prompt$read;
378 6          val = 0;
379 6      end;
380 5      end;
381 4      end;
382 3      end;
383 2      call crlf;
384 2      end get$param;

385 1      get$seg:
          procedure(type,record$size) byte public;

386 2          declare (type,k,seg$no) byte;
387 2          declare (record$size,max$attr) address;

388 2          if not bnk$swt then
389 2              return 0;

390 2          seg$no = 0ffh;
391 2          max$attr = 0ffffh;
392 2          do k = 1 to num$seg;
393 3              if mem$tbl(k).attr >= record$size then
394 3                  if type = 1 then
395 3                      do;
396 4                          if (mem$tbl(k).bank = 0) and
                              (mem$tbl(k).attr < max$attr) then
397 4                              do;
398 5                                  seg$no = k;
399 5                                  max$attr = mem$tbl(k).attr;
400 5                              end;
401 4                          end;
402 3                      else
403 4                          do;
404 4                              if (mem$tbl(k).bank <> 0) and
                                  (mem$tbl(k).attr < max$attr) then
405 5                                  do;
406 5                                      seg$no = k;
407 5                                      max$attr = mem$tbl(k).attr;
408 5                                  end;
409 3                              end;
410 2                          end;
          if (seg$no = 0ffh) and (type = 2) then

```

CP/M-PLUS V3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # CP3-135

```

411 2      do k = 1 to num$seg;
412 3          if (mem$tbl(k).attr >= record$size) and
              (mem$tbl(k).bank = 0)          and
              (mem$tbl(k).attr < max$attr)    then
413 3              do;
414 4                  seg$no = k;
415 4                  max$attr = mem$tbl(k).attr;
416 4              end;
417 3          end;
418 2      return seg$no;

419 2      end get$seg;

420 1  plm:
      procedure public;

421 2      st$ascii$hex:
      procedure(string$adr,val);
422 3          declare string$adr address;
423 3          declare string based string$adr (6) byte;
424 3          declare val address;
425 3          declare i byte;
426 3          string(0) = ' ';
427 3          string(1) = ' ';
428 3          string(2) = hexASCII(shr(high(val),4));
429 3          string(3) = hexASCII(high(val) and 0fh);
430 3          string(4) = hexASCII(shr(low(val),4));
431 3          string(5) = hexASCII(low(val) and 0fh);
432 3      end st$ascii$hex;

433 2      setup$scb:
      procedure;
434 3          declare scb$adr address;
435 3          declare scb$dat based scb$adr (100) byte;

436 3          scb$adr = .memory + shl(double(scb$pg-res$pg),8) + 09ch;

437 3          scb$dat(13h) = bdrive;
438 3          scb$dat(1ah) = con$wid;
439 3          scb$dat(1ch) = con$pag;
440 3          scb$dat(2eh) = bck$spc;
441 3          scb$dat(2fh) = rubout;
442 3          call movef(5,.(012h,07h,0,0,0),.scb$dat(58h)); /* December 15, 1982 */
443 3          if not lerror then
444 3              scb$dat(57h) = scb$dat(57h) and 7fh;
445 3          if not dbl$alv and not bnk$swt then
446 3              scb$dat(57h) = scb$dat(57h) or 0100$0000B;
          else
447 3              scb$dat(57h) = scb$dat(57h) and 1011$1111B;
448 3          scb$dat(5eh) = bnk$top;

449 3      end setup$scb;

450 2      get$drv$tbl$adr:
      procedure address;
451 3          declare temp$adr address;
452 3          declare temp2 based temp$adr address;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

453 3      declare temp3 address;

454 3      temp$adr = .memory(43h);
455 3      temp3 = temp2 + 1 + .memory;
456 3      temp$adr = temp3;
457 3      if temp2 = 0ffffh
458 3      then res$flg = 2;
459 3      else res$flg = 0;
460 3      if temp2 < 0ffffh
461 3      then return temp2 + .memory;
462 3      else return 0ffffh;
463 3      end get$drvtbl$adr;

464 2      page$chop:
465 3      procedure;
466 3      declare i byte;

467 3      drvtbl$adr = get$drvtbl$adr;

468 3      dont$hash = true;
469 3      if (drvtbl$adr <> 0ffffh) then
470 4          do;
471 5              do i = 0 to 15;
472 5                  if drvtbl(i) <> 0 then
473 6                      do;
474 6                          dph$adr = drvtbl(i) + .memory;
475 6                          if dph$hash <> 0ffffh then
476 6                              dont$hash = false;
477 5                      end;
478 4                  if dont$hash and not bnk$swt then
479 4                      res$flg = 2;
480 4                  else
481 4                      res$flg = 0;
482 3              end;

483 2      end page$chop;

484 2      get$xmove:
485 3      procedure boolean;
486 3      declare xmove$adr address;
487 3      declare xmove$val based xmove$adr byte;

488 3      call movef(2,.memory(58h),.xmove$adr);
489 3      xmove$adr = xmove$adr + .memory;
490 3      if xmove$val = 0c9h /* ret instr. */ then
491 3          return false;
492 3      else
493 3          return true;
494 3      end get$xmove;

495 2      display$layout:
496 3      procedure(string$adr,base,length);
497 3      declare string$adr address;
498 3      declare base address;
499 3      declare length byte;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

496 3      call print$console$buffer (,(cr,lf,' ','$'));
497 3      call print$console$buffer (string$adr);
498 3      call write$console(' ');
499 3      call dsply$hex$adr (base);
500 3      call write$console(' ');
501 3      call dsply$hex$high$adr (length);
502 3      if prt$msg then
503 3          do;
504 4          call movef(12,string$adr,.system$data(prt$msg$ptr));
505 4          prt$msg$ptr = prt$msg$ptr + 12;
506 4          call st$ascii$hex(.system$data(prt$msg$ptr),base);
507 4          prt$msg$ptr = prt$msg$ptr + 6;
508 4          call st$ascii$hex(.system$data(prt$msg$ptr),
                                double(length)*256);
509 4          prt$msg$ptr = prt$msg$ptr + 6;
510 4          call movef(3,(cr,lf,' '),.system$data(prt$msg$ptr));
511 4          prt$msg$ptr = prt$msg$ptr + 3;
512 4      end;
513 3      end display$layout;

514 2      reloc$module;
          procedure (fcb$adr);
515 3          declare fcb$adr address;
516 3          FCBin = fcb$adr;
517 3          if relfix <> 0 then
518 3              do;
519 4              call error(true,1,('Disk read error: $'));
520 4              end;
521 3          call close$file(fcb$adr);
522 3          end reloc$module;

523 2      load;
          procedure (fcb$adr,atts$adr);
524 3          declare fcb$adr address;
525 3          declare atts$adr address;
526 3          declare atts based atts$adr (3) address;
527 3          declare (i,rdcnt) byte;

528 3          prgsiz = atts(0);
529 3          bufsiz = atts(1);
530 3          codsiz = atts(2);
531 3          call movef(8,fcb$adr+1,.fcb$msg);
532 3          call movef(3,fcb$adr+9,.fcb$msg+9);
533 3          if shr(prgsiz+255,7) > nmb$sect then
534 3              do;
535 4              call error(true,1,('File cannot fit into GENCPM buffer: ','$'));
536 4              end;
537 3          rdcnt = low(shr(prgsiz-1,7)) + 1;
538 3          i = 0;
539 3          do while (i < rdcnt);
540 4              call set$dma$address(dma:=.sctbfr(i));
541 4              call read$record(fcb$adr);
542 4              i = i + 1;
543 4          end;
544 3          call movef(128,dma,.bitmap); /* copy the last sector read, into */
                                         /* the bitmap buffer, relocation */
                                         /* info might be that last sector */

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

545 3      dma = prgsiz + .memory;
546 3      end load;

547 2      wrtbuf:
          procedure (wrtlen,wrtoff$adr);
548 3          declare (i,wrtlen,wrtcnt) byte;
549 3          declare wrtoff$adr address;
550 3          declare wrtoff based wrtoff$adr address;
551 3          if wrtlen <> 0 then
552 3              do;
553 4                  call movef(8,.FCBout+1,.fcb$msg);
554 4                  call movef(3,.FCBout+9,.fcb$msg+9);
555 4                  FCBout(33) = low(wrtoff);
556 4                  FCBout(34) = high(wrtoff);
557 4                  call write$random$record(.FCBout);
558 4                  dma = dma + low(256 - low(dma - .memory));
559 4                  wrtcnt = wrtlen * 2 - 1;
560 4                  do i = 0 to wrtcnt;
561 5                      call set$dma$address(dma:=dma-80h);
562 5                      call write$record(.FCBout);
563 5                  end;
564 4                  call set$random$record(.FCBout);
565 4                  call movef(2,.FCBout(33),wrtoff$adr);
566 4              end;
567 3          end wrtbuf;

568 2      get$file$info:
          procedure;
569 3          declare fcb$adr address;
570 3          declare atts$adr address;
571 3          declare file$atts based atts$adr(3) address;
572 3          declare header$record structure (
              fill1 byte,
              psize address,
              fill2 byte,
              dsize address,
              fill3 (4) byte,
              csize address,
              fill4 (116) byte) at (.memory);

573 3      get$atts:
          procedure;
574 4          call movef(8,fcb$adr+1,.fcb$msg);
575 4          call movef(3,fcb$adr+9,.fcb$msg+9);
576 4          if open$file(fcb$adr) = 0ffh
              then call error(true,1,('Unable to open: $'));
          call set$dma$address(header$record);
          call read$record(fcb$adr);
          file$atts(0) = header$record.psize;
          file$atts(1) = header$record.dsize;
          file$atts(2) = header$record.csize;
          call read$record(fcb$adr);
          end get$atts;

585 3      if not bnk$swt then
586 3          do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

587 4      call movef(8,('BDOS3  '),.res$fcbl);
588 4      call movef(8,('BIOS3  '),.bios$fcbl);
589 4      end;
        else
590 3      do;
591 4          fcb$adr = .bnk$fcbl;
592 4          atts$adr = .bnk$atts;
593 4          call get$atts;
594 4      end;
595 3      fcb$adr = .bios$fcbl;
596 3      atts$adr = .bios$atts;
597 3      call get$atts;
598 3      fcb$adr = .res$fcbl;
599 3      atts$adr = .res$atts;
600 3      call get$atts;
601 3      end get$file$info;

602 2      need$tbl:
        procedure byte;
603 3          declare (all$some,i) byte;

604 3          all$some = false;

605 3          if drvtbl$adr = 0ffffh
            then return false;
        else
607 3          do i = 0 to 15;
608 4              if drvtbl(i) <> 0 then
609 4                  do;
610 5                      dph$adr = drvtbl(i) + .memory;
                        /* zero the reserved bytes in the DPH */
611 5                      call movef(9,(0,0,0,0,0,0,0,0,0),dph$adr+2);
612 5                      if (dph.dirbcb = 0ffffh) or (dph.dtabcb = 0ffffh) or
                        (dph.hash = 0ffffh) or (dph.alv = 0ffffh) or
                        (dph.csv = 0ffffh)
                        then all$some = true;
                    end;
614 5          end;
615 4          end;
616 3          return all$some;

617 3      end need$tbl;

618 2      setup$hash:
        procedure;
619 3          declare (i,j,printed,seg$no,seg0$no,h$bank,h$hash) byte;
620 3          declare (size,h$attr,max$attr,max0$attr) address;
621 3          declare nohash boolean;

622 3          printed = false;
623 3          nohash = true;

624 3          do i = 0 to 15;
625 4              dph$adr = drvtbl(i) + .memory;
626 4              if drvtbl(i) <> 0 then
627 4                  do;
628 5                      if dph.hash < 0ffffh then
629 5                          nohash = false;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 570

PACIFIC GROVE, CA 93950

SER. # _____

```

630 5      if dph.hash = 0ffffh then
631 5          do;
632 6              if not printed then
633 6                  do;
634 7                      printed = true;
635 7                      call print$console$buffer(
                          (if,cr,'Setting up directory hash tables!',
                           lf,cr,'$'));
636 7                      end;
637 6                      query = quest(27 + i);
638 6                      dph$adr = dph.dpb + .memory;
639 6                      size = shl(dpb.drm+1,2);
640 6                      call print$console$buffer(
                          (' Enable hashing for drive $'));
641 6                      call write$console('A'+i);
642 6                      call print$console$buffer((':' $));
643 6                      call get$response(.hash(i));
644 6                      call crlf;
645 6                      if not hash(i) then
646 6                          do;
647 7                              dph.hash = 0ffffh;
648 7                              end;
649 6                      else
650 6                          if not bnk$swt then
651 7                              do;
652 7                                  nohash = false;
653 7                                  hash$data(i) = size;
654 7                                  hash$space = hash$space + size;
655 7                                  end;
656 6                          else
657 7                              do;
658 7                                  if (seg$no := get$seg(2,size)) = 0ffh then
659 8                                      call error(false,0,('
660 8                                          Unable to allocate space for hash table.$'));
661 8                                  else
662 8                                      do;
663 7                                          do;
664 6                                              dph.hbank = mem$tbl(seg$no).bank;
665 6                                              dph.hash = shl(double(mem$tbl(seg$no).base),8) +
666 6                                                  (shl(double(mem$tbl(seg$no).len),8) -
667 6                                                      mem$tbl(seg$no).attr);
668 8                                              mem$tbl(seg$no).attr = mem$tbl(seg$no).attr - size;
669 8                                          end;
670 7                                          end;
671 6                                          end;
672 5                                          end;
673 4                                          end;
674 3                                          if (not bnk$swt) and (nohash) then
675 3                                              do;
676 4                                                  res$flg = 2;
677 4                                                  scb$pg = scb$pg + 2;
678 4                                                  res$pg = res$pg + 2;
679 4                                                  end;
680 3                                              end setup$hash;
681 2
682 2      get$alloc$chk;
683 2      procedure;
684 3      declare (i,dbl$alloc) byte;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

676 3      declare printed boolean;

677 3      do i = 0 to 15;
678 4          alloc(i) = 0;
679 4          chk(i) = 0;
680 4      end;

681 3      if not dbl$alv and not bnk$swt then
682 3          dbl$alloc = 1;
        else
683 3          dbl$alloc = 2;

684 3      alloc$space = 0;
685 3      chk$space = 0;
686 3      printed = false;
687 3      do i = 0 to 15;
688 4          if drvtbl(i) <> 0 then
689 4              do;
690 5                  dph$adr = drvtbl(i) + .memory;
691 5                  dpb$adr = dph,dpb + .memory;
692 5                  if dph,alv = 0fffeh then
693 5                      do;
694 6                          call print$console$buffer(.(cr,lf,
                              'Setting up Allocation vector for drive $'));
695 6                          call write$console('A'i);
696 6                          call write$console(':');
697 6                          printed = true;
698 6                          alloc(i) = (dpb,dsm/8 + 1) * dbl$alloc;
699 6                          alloc$space = alloc$space + alloc(i);
700 6                      end;
701 5                  if dph,csv = 0fffeh then
702 5                      do;
703 6                          call print$console$buffer(.(cr,lf,
                              'Setting up Checksum vector for drive $'));
704 6                          call write$console('A'i);
705 6                          call write$console(':');
706 6                          printed = true;
707 6                          chk(i) = (dpb,dsm + 4)/4;
708 6                          chk$space = chk$space + chk(i);
709 6                          dpb,cks = (dpb,cks and 8000h) or chk(i);
710 6                      end;
711 5                  end;
712 4              end;
713 3          if printed then
714 3              call crlf;
715 3      end get$alloc$chk;

716 2      setup$mem$seg$tbl:
        procedure;
717 3          declare (i,j,ok,accept,mlo,high,tlow,thigh) byte;
718 3          declare mem$temp address;

        /* Create first memory segment table entry */
719 3          mem$tbl(0).base = bnk$pg;
720 3          mem$tbl(0).len = bnk$top - bnk$pg;
721 3          mem$tbl(0).attr = 0;
722 3          mem$tbl(0).bank = 0;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

723 3      accept = false;
724 3      call print$console$buffer(
              .(lf,cr,
                '*** Bank 1 and Common are not included ***',
                lf,cr,
                '*** in the memory segment table.      ***',
                lf,cr,lf,cr,'$'));
725 3      query = quest(10);
726 3      call get$param (.( 'Number of memory segments $'),
              .num$seg,10);
727 3      call print$console$buffer(.(cr,lf,
              'CP/M 3 Base,size,bank ($'));
728 3      call dsply$hex(mem$tbl(0).base);
729 3      call write$console(' ');
730 3      call dsply$hex(mem$tbl(0).len);
731 3      call write$console(' ');
732 3      call dsply$hex(mem$tbl(0).bank);
733 3      call print$console$buffer(.( ' ') ,lf,cr,'$'));
734 3      do while not accept;
              /* Bank switched memory segment table input */
735 4      call print$console$buffer (.(cr,lf,
              'Enter memory segment table:',lf,cr,'$'));
736 4      do j = 1 to num$seg;
737 5          ok = false;
738 5          do while not ok;
739 6              query = quest(11 + j - 1);
740 6              call get$param (.( ' Base,size,bank ', '$'),
              .mem$tbl(j),16);
741 6              mem$tbl(j).attr = shl(double(mem$tbl(j).len),8);
742 6              if mem$tbl(j).len = 0 then
743 6                  do;
744 7                  call error(false,0,.(
              'Zero length segment not allowed.$'));
745 7                  end;
              else
746 6                  if mem$tbl(j).bank = 1 then
747 6                      do;
748 7                      call error(false,0,.(
              'Bank one not allowed.$'));
749 7                      end;
              else
750 6                  do;
751 7                      tlow = mem$tbl(j).base;
752 7                      mem$temp = double(tlow) + double(mem$tbl(j).len);
753 7                      if (high(mem$temp) <> 0) or (low(mem$temp) > bnk$top) then
754 7                          do;
755 8                          call print$console$buffer(.(cr,lf,'ERROR: ',
              'Memory conflict - segment trimmed.',
              cr,lf,'$'));
756 8                          mem$tbl(j).len = bnk$top - tlow;
757 8                          mem$tbl(j).attr = shl(double(bnk$top - tlow),8);
758 8                          end;
              else
759 7                          do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

760 8      thigh = low(mem$temp);
761 8      i = 0;
762 8      ok = true;
763 8      do while ((i < j) and ok);
764 9          mlow = mem$tbl(i).base;
765 9          mhigh = mlow + mem$tbl(i).len;
766 9          if mem$tbl(i).bank = mem$tbl(j).bank then
767 9              do;
768 10                  if (mhigh >= thigh) and (tlow >= mlow) then
769 10                      do;
770 11                          call error(false,0,.(
                              'Memory conflict - cannot trim segment.$'));
771 11                          ok = false;
772 11                      end;
773 10                  else
774 10                      if ((thigh > mhigh) and (mhigh > tlow)) then
775 11                          do;
776 11                              call print$console$buffer(.(cr,lf,'ERROR: ',
                              'Memory conflict - segment trimmed.',
                              cr,lf,'$'));
777 11                              mem$tbl(j).base = mhigh;
778 11                              ok = false;
779 10                          end;
780 10                      else
781 11                          if ((thigh > mlow) and (mlow > tlow)) then
782 11                              do;
783 11                                  call print$console$buffer(.(cr,lf,'ERROR: ',
784 11                                  'Memory conflict - segment trimmed.',
785 11                                  cr,lf,'$'));
786 11                                  mem$tbl(j).len = mlow - tlow;
787 11                                  mem$tbl(j).attr = shl(double(mlow-tlow),8);
788 11                                  ok = false;
789 11                              end;
790 11                          end;
791 11                      end;
792 11                      i = i + 1;
793 11                  end;
794 11              end;
795 11          end;
796 11      end;
797 11      call crlf;
798 11      do j = 0 to num$seg;
799 11          if j = 0 then
800 11              call print$console$buffer (.( ' CP/M 3 Sys ', '$'));
801 11          else
802 11              do;
803 11                  call print$console$buffer (.( ' Memseg No. ', '$'));
804 11                  call dsply$hex(j-1);
805 11              end;
806 11              call dsply$hex$high$adr (mem$tbl(j).base);
807 11              call dsply$hex$high$adr (mem$tbl(j).len);
808 11              if bnk$swt then
809 11                  do;
810 11                      call print$console$buffer (.( ' Bank ', '$'));
811 11                      call dsply$hex (mem$tbl(j).bank);
812 11                  end;
813 11              call crlf;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

809 5      end;
810 4      query = false;
811 4      accept = true;
812 4      call print$console$buffer (.(cr,lf,
      'Accept new memory segment table entries ','$'));
813 4      call get$response (.(accept));
814 4      end; /* do while not accept */
815 3      call crlf;
816 3      end setup$mem$seg$tbl;

817 2      get$default$file:
      procedure;
818 3      declare ret byte;

819 3      call print$console$buffer(.(
      'Default entries are shown in (parens).',cr,lf,
      'Default base is Hex, precede entry with $ for decimal',
      cr,lf,'$'));
820 3      if (ret:=open$file(.data$fcbl)) <> 255 then
821 3      do;
822 4          call movef(8,.data$fcbl,.fcb$msg);
823 4          call movef(3,.data$fcbl+9,.fcb$msg+9);
824 4          call print$console$buffer(.(
      cr,lf,'Use GENCPM.DAT for defaults $'));
825 4          ret = 0ffh;
826 4          call get$response(.ret);
827 4          call crlf;
828 4          if ret then
829 4              call getdef;
830 4          call close$file(.data$fcbl);
831 4          end;
      else
832 3      do;
833 4          display = true;
834 4          automatic = false;
835 4          end;

836 3      end get$default$file;

837 2      setup$system$dat:
      procedure;
838 3      declare (i,j,ok,temp) byte;
839 3      ok = false;
840 3      call get$default$file;
841 3      do while not ok;
842 4          query = quest(155);
843 4          call crlf;
844 4          call print$console$buffer(.(('Create a new GENCPM.DAT file $'));
845 4          call get$response(.crdatf);
846 4          query = quest(0);
847 4          call crlf;
848 4          call crlf;
849 4          call print$console$buffer(.(('Display Load Map at Cold Boot $'));
850 4          call get$response(.prt$msg);
851 4          call crlf;
852 4          call crlf;
853 4          query = quest(1);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

854 4      con$wid = con$wid + 1;
855 4      call get$param (,('Number of console columns $'),
                        ,con$wid,10);
856 4      con$wid = con$wid - 1;
857 4      query = quest(2);
858 4      con$pag = con$pag + 1;
859 4      call get$param (,('Number of lines in console page $'),
                        ,con$pag,10);
860 4      con$pag = con$pag - 1;
861 4      query = quest(3);
862 4      call print$console$buffer(,
                        ('Backspace echoes erased character $'));
863 4      call get$response (,bck$spc);
864 4      call crlf;
865 4      query = quest(4);
866 4      call print$console$buffer(,
                        ('Rubout echoes erased character $'));
867 4      call get$response (,rubout);
868 4      call crlf;
869 4      call crlf;
870 4      query = quest(5);
871 4      err1:
            call print$console$buffer(,('Initial default drive ($')));
            call write$console('A'+bdrive);
            call print$console$buffer(,(':') ? $));
            call read$console$buffer(,lnbfr);
            if lnbfr(1) <> 0 then
            do;
                temp = upper(lnbfr(2))- 'A';
                if not valid$drive(temp) then
                goto err1;
                bdrive = temp;
            end;
            call crlf;
            call crlf;
            query = quest(6);
            call get$param (,('Top page of memory $'),
                        ,mem$top,16);
886 4      os$top = shl(double(mem$top),8) + 100h;
887 4      query = quest(7);
888 4      call print$console$buffer(,('Bank switched memory $'));
889 4      call get$response (,bnk$swt);
890 4      call crlf;
891 4      non$bnk = not bnk$swt;
892 4      if bnk$swt then
893 4      do;
            query = quest(8);
            call get$param (,('Common memory base page $'),
                        ,bnk$top,16);
896 5      call crlf;
897 5      query = quest(9);
898 5      call print$console$buffer(,('Long error messages $'));
899 5      call get$response(,lerror);
900 5      call crlf;
901 5      end;
            else
902 4      do;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

903 5      query = quest(156);
904 5      call crlf;
905 5      call print$console$buffer(,('Double allocation vectors $'));
906 5      call get$response(.dbl$alv);
907 5      call crlf;
908 5      bnk$top = 0;
909 5      end;
910 4      query = false;
911 4      ok = true;
912 4      call crlf;
913 4      call print$console$buffer(,('Accept new system definition $'));
914 4      call get$response(.ok);
915 4      call crlf;
916 4      end;
917 3      save$mem$top = mem$top;
918 3      mem$top = mem$top + 1;
919 3      rubout = not rubout;
920 3      end setup$system$dat;

921 2      setup$CPM80$sys:
          procedure;
922 3      declare i byte;
923 3      call print$console$buffer (, ( cr,lf,lf,
          'CP/M 3.0 System Generation',cr,lf,
          'Copyright (C) 1982, Digital Research',
          cr,lf,cr,lf,'$'));
924 3      call delete$file (.fcbout);
925 3      call create$file (.fcbout);
926 3      FCBout(32) = 0;
927 3      do i = 0 to 127;
928 4          system$data(i) = 0;
929 4      end;
930 3      do i = 128 to 255;
931 4          system$data(i) = '$';
932 4      end;
933 3      prt$msg$ptr = 128;
934 3      call movef(3, (cr,lf,' '),.system$data(prt$msg$ptr));
935 3      prt$msg$ptr = 131;
936 3      call movef(8,.FCBout+1,.fcb$msg);
937 3      call movef(3,.FCBout+9,.fcb$msg+9);
938 3      call set$DMA$address (.sctbfr);
939 3      call write$record (.FCBout);
940 3      call write$record (.FCBout);
941 3      end setup$CPM80$sys;

942 2      initialization:
          procedure;
943 3      declare i byte;

944 3      nmb$sect = shr ((maxb-.sctbfr+1),7);

945 3      do i = 0 to 15;
946 4          hash$data(i) = 0;
947 4      end;
948 3      hash$space = 0;

949 3      if fcb(1) = 'A' then

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

950 3      do;
951 4          automatic = true;
952 4          display = false;
953 4          do i = 0 to 154;
954 5              quest(i) = false;
955 5          end;
956 4      end;
          else
957 3          do;
958 4              automatic = false;
959 4              display = true;
960 4          end;
961 3          if fcb16(1) = 'D' then
962 3              do;
963 4                  display = true;
964 4              end;
965 3          query = false;
966 3      end initialization;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

/*
  GENCPM Main Program
*/

```

```

967 2      res$flg = 0;
968 2      display = true;
969 2      call setup$CPM80$sys;
970 2      call initialization;
971 2      call setup$system$dat;

972 2      call get$file$info;

973 2      if bios$atts(2) <> 0
          then res$bios$len = high(bios$atts(2) + 255);
975 2      else res$bios$len = high(bios$atts(0) + 255);
976 2      bios$pg = nem$top - res$bios$len;
977 2      bnk$off = bnk$top - (high(bios$atts(0) + 255) - res$bios$len);
978 2      bnk$pg = bnk$off - high(bnk$atts(0) + 255);

979 2      call load(.bios$fcb,.bios$atts);

980 2      call page$chop;

981 2      if not bnk$swt then
982 2          do;
983 3          scb$pg = bios$pg - (3 - res$flg);
984 3          res$pg = bios$pg - high(res$atts(0) + 255) + res$flg;
985 3          end;
          else
986 2          do;
987 3          scb$pg = bios$pg - 1;
988 3          res$pg = bios$pg - high(res$atts(0) + 255);
989 3          end;

990 2      if need$tbl then

```

```

991 2      do;
992 3          call get$alloc$chk;
993 3          if bnk$swt then
994 3              do;
995 4                  bnk$off = bnk$top - (high(bios$atts(0) + alloc$space +
996 4                      chk$space + 255) - res$bios$len);
997 4                  bnk$pg = bnk$off - high(bnk$atts(0) + 255);
998 4                  xmove$implemented = get$xmove;
999 4                  call setup$mem$seg$tbl;
1000 4                  if (not xmove$implemented) then
1001 5                      do len = 0 to 15;
1002 5                          record(len).altbnks = false;
1003 4                      end;
1004 3                  else
1005 3                      xmove$implemented = false;
1006 3                      if not dont$hash then
1007 3                          call setup$hash;
1008 3                          call setbuf;
1009 2      res$len = res$bios$len;
1010 2      offset = bios$pg;
1011 2      call reloc$module(,bios$fcb);
1012 2      if bnk$swt
1013 3          then call display$layout(,('BNKBIOS3 SPR$'),
1014 3              double(bios$pg)*256,res$bios$len);
1015 2      else call display$layout(,('BIOS3 SPR$'),
1016 3              double(bios$pg)*256,res$bios$len);
1017 2      if not bnk$swt then
1018 2          do;
1019 3              len = res$bios$len;
1020 3              off = 2;
1021 3              call wrtbuf(len,,off);
1022 3              common$len = len;
1023 3              banked$len = 0;
1024 2          end;
1025 2      else
1026 2          do;
1027 3              len = high(bios$atts(0) + 255) - res$bios$len;
1028 3              off = (high(res$atts(0) + 255) + res$bios$len) * 2 + 2;
1029 3              call display$layout(,('BNKBIOS3 SPR$'),double(bnk$off)*256,len);
1030 3              call wrtbuf(len,,off);
1031 3              banked$len = len;
1032 3              len = res$bios$len;
1033 3              off = 2;
1034 3              dma = dma - 80h;
1035 3              call wrtbuf(len,,off);
1036 3              common$len = len;
1037 2          end;
1038 2      res$len = high(res$atts(0) + 255) - res$flg;
1039 2      offset = res$pg;
1040 2      call load(,res$fcb,,res$atts);
1041 2      call reloc$module(,res$fcb);

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

1039 2    call setup$scb;
1040 2    dma = dma - (res$flg * 256);
1041 2    len = high(res$atts(0) + 255) - res$flg;
1042 2    if not bnk$swt
        then call display$layout(,('BDOS3   SPR$'),double(res$pg)*256,len);
1044 2    else call display$layout(,('RESBDOS3 SPR$'),double(res$pg)*256,len);
1045 2    call wrtbuf(len,.off);
1046 2    common$len = common$len + len;

1047 2    if bnk$swt then
1048 2    do;
1049 3        res$len = 0ffh;
1050 3        offset = bnk$pg;
1051 3        call load(,bnk$fcb,.,bnk$atts);
1052 3        call reloc$module(,bnk$fcb);
1053 3        len = high(bnk$atts(0) + 255);
1054 3        off = off + (high(bios$atts(0) + 255) - res$bios$len) * 2;
1055 3        call display$layout(,('BNKBDOS3 SPR$'),double(bnk$pg)*256,len);
1056 3        call wrtbuf(len,.off);
1057 3        banked$len = banked$len + len;
1058 3    end;
1059 2    if not prt$msg then prt$msg$ptr = prt$msg$ptr - 3;
1061 2    call movef(12,.(lf,cr,' 64K TPA',lf,cr),.system$data(prt$msg$ptr));
1062 2    res$pg = shr(res$pg,2);
1063 2    system$data(prt$msg$ptr+3) = res$pg/10 + '0';
1064 2    system$data(prt$msg$ptr+4) = res$pg mod 10 + '0';
1065 2    prt$msg$ptr = prt$msg$ptr + 12;
1066 2    sys$entry = bios$pg * 256;
1067 2    call movef(36,('Copyright (C) 1982, Digital Research'),.system$data(10h));
1068 2    call movef(6,.memory,.system$data(35h)); /* Copy Serial No. into header */
1069 2    FC Bout(33) = 0; FC Bout(34) = 0; FC Bout(35) = 0;
1072 2    system$data(0) = mem$top;
1073 2    system$data(2) = bnk$top;
1074 2    call movef(8,.FC Bout+1,.,fcb$msg);
1075 2    call movef(3,.,FC Bout+9,.,fcb$msg+9);
1076 2    call set$DMA$address(,system$data);
1077 2    call write$random$record(,FC Bout);
1078 2    FC Bout(33) = 1;
1079 2    call set$DMA$address(,system$data(128));
1080 2    call write$random$record(,FC Bout);
1081 2    call close$file(,fcbout);

1082 2    if crdatf then
1083 2    do; /* create a new data file for GENCPM */
1084 3        crdatf = false;
1085 3        mem$top = save$mem$top;
1086 3        rubout = not rubout;
1087 3        call movef(8,.data$fcb+1,.,fcb$msg);
1088 3        call movef(3,.,data$fcb+9,.,fcb$msg+9);
1089 3        call crtdef;
1090 3    end;

1091 2    display = true;
1092 2    call print$console$buffer (,.(cr,lf,lf,
        '*** CP/M 3.0 SYSTEM GENERATION DONE ***', '$'));
1093 2    return;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

1094 2 end plm;
1095 1 end gencpm;

MODULE INFORMATION:

CODE AREA SIZE = 2362H 9058D
VARIABLE AREA SIZE = 0302H 770D
MAXIMUM STACK SIZE = 000EH 14D
1478 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

CP/M PLUS V3.0
COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CREATEDefaults
 OBJECT MODULE PLACED IN CRDEF.OBJ
 COMPILER INVOKED BY: PLM80 CRDEF.PLN DEBUG OPTIMIZE

```

1      $title('GENCPM Token File Creator')
      create$defaults:
      do;

      /*
        Copyright (C) 1982
        Digital Research
        P.O. Box 579
        Pacific Grove, CA 93950
      */

      /*
        Revised:
        20 Sept 82 by Bruce Skidmore
      */

2 1      declare true literally 'OFFH';
3 1      declare false literally '0';
4 1      declare forever literally 'while true';
5 1      declare boolean literally 'byte';
6 1      declare cr literally '0dh';
7 1      declare lf literally '0ah';
8 1      declare tab literally '09h';

      /*
        Data Structures
      */

9 1      declare data$fcbl (36) byte external;
10 1     declare obuf (128) byte at (.memory);
11 1     declare hexASCII (16) byte external;
12 1     declare symbtbl (20) structure(
          token(8) byte,
          len      byte,
          flags    byte,
          qptr     byte,
          ptr      address) external;

      /*
        B D O S   P r o c e d u r e   &   F u n c t i o n   C a l l s
      */

13 1     delete$file:
          procedure (fcb$address) external;
14 2         declare fcb$address address;
15 2         end delete$file;

16 1     create$file:

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

17 2      procedure (fcb$address) external;
18 2      declare fcb$address address;
        end create$file;

19 1      close$file;
        procedure (fcb$address) external;
20 2      declare fcb$address address;
21 2      end close$file;

22 1      write$record;
        procedure (fcb$address) external;
23 2      declare fcb$address address;
24 2      end write$record;

25 1      set$DMA$address;
        procedure (DMA$address) external;
26 2      declare DMA$address address;
27 2      end set$DMA$address;

/*
   Main CRTDEF Procedure
*/
28 1      crtdef:
        procedure public;
29 2      declare (flags,symbol$done,i,j,obuf$index,inc) byte;
30 2      declare val$adr address;
31 2      declare val based val$adr byte;

32 2      inc$obuf$index:
        procedure;

33 3          if obuf$index = 7fh then
34 3          do;
35 4              call write$record(.data$fcb);
36 4              do obuf$index = 0 to 7fh;
37 5                  obuf(obuf$index) = lah;
38 5              end;
39 4              obuf$index = 0;
40 4          end;
        else
41 3          obuf$index = obuf$index + 1;

42 3      end inc$obuf$index;

43 2      emit$ascii$hex:
        procedure(dig);
44 3      declare dig byte;

45 3          call inc$obuf$index;
46 3          obuf(obuf$index) = hexASCII(shr(dig,4));
47 3          call inc$obuf$index;
48 3          obuf(obuf$index) = hexASCII(dig and 0fh);

49 3      end emit$ascii$hex;

50 2      call set$dma$address(.obuf);
51 2      call delete$file(.data$fcb);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

52  2      call create$file(,data$fcbl);
53  2      obuf$index = 0ffh;
54  2      do i = 0 to 21;
55  3          symbol$done = false;
56  3          flags = symb1(i).flags;
57  3          inc = 0;
58  3          do while (inc < 16) and (not symbol$done);
59  4              do j = 0 to 7;
60  5                  call inc$obuf$index;
61  5                  obuf(obuf$index) = symb1(i).token(j);
62  5              end;
63  4          if (flags and 8) = 0 then
64  4              symbol$done = true;
65  4          else
66  5              do;
67  5                  if (flags and 10h) <> 0 then
68  5                      obuf(obuf$index) = 'A' + inc;
69  5                  else
70  5                      do;
71  6                          if inc < 10 then
72  6                              do;
73  7                                  obuf(obuf$index) = '0' + inc;
74  7                              end;
75  7                          else
76  7                              do;
77  6                                  obuf(obuf$index) = 'A' + inc - 10;
78  6                              end;
79  6                          end;
80  6                      end;
81  5              end;
82  5          end;
83  5          call inc$obuf$index;
84  5          obuf(obuf$index) = ' ';
85  5          call inc$obuf$index;
86  5          obuf(obuf$index) = '=';
87  5          call inc$obuf$index;
88  5          obuf(obuf$index) = ' ';
89  5          call inc$obuf$index;
90  5          obuf(obuf$index) = ' ';
91  4          val$adr = symb1(i).ptr + (inc * symb1(i).len);
92  4          if (flags and 1) <> 0 then
93  5              do;
94  5                  call inc$obuf$index;
95  5                  obuf(obuf$index) = 'A' + val;
96  5              end;
97  4          else
98  5              do;
99  5                  if (flags and 2) <> 0 then
100 6                      do;
101 7                          call inc$obuf$index;
102 7                          if val then
103 7                              obuf(obuf$index) = 'Y';
104 7                          else

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SFR. #

```

96  6          obuf(obuf$index) = 'N';
97  6          end;
          else
98  5          do;
99  6          call emit$ascii$hex(val);
100  6          if (flags and 18h) = 8 then
101  6          do;
102  7          call inc$obuf$index;
103  7          obuf(obuf$index) = ',';
104  7          val$adr = val$adr + 1;
105  7          call emit$ascii$hex(val);
106  7          call inc$obuf$index;
107  7          obuf(obuf$index) = ',';
108  7          val$adr = val$adr + 1;
109  7          call emit$ascii$hex(val);
110  7          end;
111  6          end;
112  5          end;

113  4          call inc$obuf$index;
114  4          obuf(obuf$index) = cr;
115  4          call inc$obuf$index;
116  4          obuf(obuf$index) = lf;

117  4          inc = inc + 1;

118  4          end;

119  3          end;

120  2          if obuf$index <= 7fh then
121  2          call write$record(.data$fcb);
122  2          call close$file(.data$fcb);

123  2          end crtdef;
124  1          end create$defaults;
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93350

SER. # _____

MODULE INFORMATION:

CODE AREA SIZE = 0298H 664D
VARIABLE AREA SIZE = 0009H 9D
MAXIMUM STACK SIZE = 0006H 6D
201 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

LOC OBJ SEQ SOURCE STATEMENT

```

1 $title ('CP/M V3.0 Relocate and Fix Up File')
2     name    relfix
3 ;
4 ;/*
5 ; Copyright (C) 1979,1980,1981,1982
6 ; Digital Research
7 ; P.O. Box 579
8 ; Pacific Grove, CA 93950
9 ;
10 ; Revised:
11 ;   05 Aug 82 by Bruce Skidmore
12 ;*/
13
14     cseg
15
16     extrn  mon1    ;BDOS entry point
17     extrn  FCBin   ;FCB for input
18     extrn  scbfr    ;sector buffer
19     extrn  offset   ;relocation offset
20     extrn  prgsiz   ;program size
21     extrn  bufsiz   ;buffer size
22     extrn  bnkpg    ;bnkbdos page
23     extrn  respg    ;resbdos page
24     extrn  scbpg    ;System Control Block page
25     extrn  biospg   ;Bios page
26     extrn  reslen   ;Resident System length
27     extrn  bnkoff   ;Banked System offset
28     extrn  nonbnk   ;Non Banked CP/M flag
29
30     public bitmap   ;bitmap buffer
31
32 RelFix:
33     public RelFix
34     lxi    d,bitmap
35     mvi    c,26
36     call   mon1      ;set DMA address to bit map
37 ;
38     ;file loaded, ready for relocation
39     lhld   prgsiz
40     mov    b,h
41     mov    c,l        ;BC = program size
42     mov    a,l
43     ani    127
44     mov    l,a
45     jnz    nofill      ;if program size is an even number
46     push   h            ;of sectors prefill the bitmap buffer
47     push   b
48     lhld   fcbn
49     xchg
50     mvi    c,20
51     call   mon1
52     pop    b

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

LOC	OBJ	SEQ	SOURCE STATEMENT
0020	E1	53	pop h
0021	B7	54	ora a
0022	C2E300	55	jnz errtn
		56	nofill:
0025	5D	57	mov e,l ;L = offset into bitmap buffer
0026	1600	58	mvi d,0
0028	21E900	59	lxi h,bitmap
0028	19	60	dad d ;HL = bit map base
002C	3E69	61	mvi a,low(bitmap+128)
002E	326901	62	sta btmap ;save number of relocation bytes
		63	;in left in bitmap buffer
0031	110000	64	lxi d,sctbfr ;DE = base of program
0034	E5	65	push h ;save bit map base in stack
0035	3A0000	66	lda offset
0038	67	67	mov h,a ;H = relocation offset
		68	pgrel0:
0039	78	69	mov a,b ;bc=0?
003A	B1	70	ora c
003B	CADD00	71	jz ExitRelFix
		72 ;	
		73 ;	not end of the relocation,
		74 ;	may be into next byte of bit map
003E	0B	75	dcx b ;count length down
003F	7B	76	mov a,e
0040	D600	77	sui low(sctbfr)
0042	E607	78	ani 111b ;0 causes fetch of next byte
0044	C26700	79	jnz pgrel3
		80 ;	fetch bit map from stacked address
0047	E3	81	xthl
0048	3A6901	82	lda btmap
004B	BD	83	cmp l
004C	C26300	84	jnz pgrel2
004F	C5	85	push b
0050	D5	86	push d
0051	2A0000	87	lhld FCBin
0054	EB	88	xchg
0055	0E14	89	mvi c,20
0057	CD0000	90	call mon1
005A	D1	91	pop d
005B	C1	92	pop b
005C	21E900	93	lxi h,bitmap
005F	B7	94	ora a
0060	C2E300	95	jnz errtn ;return with error condition
		96	pgrel2:
0063	7E	97	mov a,m ;next 8 bits of map
0064	23	98	inx h
0065	E3	99	xthl ;base address goes back to stack
0066	6F	100	mov l,a ;l holds map as 8 bytes done
		101	pgrel3:
0067	7D	102	mov a,l
0068	17	103	ral ;cy set to 1 if reloc necessary
0069	6F	104	mov l,a ;back to l for next time around
006A	D2D900	105	jnc pgrel4 ;skip relocation if cy=0
		106 ;	
		107 ;	current address requires relocation

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

LOC	OBJ	SEQ	SOURCE STATEMENT
		108 ;	
006D	E5	109	push h
006E	1A	110	ldax d
006F	3C	111	inr a
0070	C27900	C 112	jnz test2
0073	3A0000	E 113	lda biospg
0076	C3D700	C 114	jmp endt
		115 test2:	
0079	3C	116	inr a
007A	C28B00	C 117	jnz test3
007D	3A0000	E 118	lda scbpg
0080	F5	119	push psw
0081	1B	120	dcx d
0082	1A	121	ldax d
0083	C69C	122	adi 09ch
0085	12	123	stax d
0086	13	124	inx d
0087	F1	125	pop psw
0088	C3D700	C 126	jmp endt
		127 test3:	
008B	3C	128	inr a
008C	C29500	C 129	jnz test4
008F	3A0000	E 130	lda respg
0092	C3D700	C 131	jmp endt
		132 test4:	
0095	3C	133	inr a
0096	C29F00	C 134	jnz test5
0099	3A0000	E 135	lda bnkpg
009C	C3D700	C 136	jmp endt
		137 test5:	
009F	3C	138	inr a
00A0	C2A900	C 139	jnz test6
00A3	3A0000	E 140	lda scbpg
00A6	C3D700	C 141	jmp endt
		142 test6:	
00A9	3A0000	E 143	lda reslen
00AC	67	144	mov h,a
00AD	3A0000	E 145	lda nonbnk
00B0	B7	146	ora a
00B1	CAC400	C 147	jz test7
00B4	1A	148	ldax d
00B5	94	149	sub h
00B6	DAD100	C 150	jc default
00B9	1B	151	dcx d
00BA	3E9A	152	mvi a,09ah
00BC	12	153	stax d
00BD	13	154	inx d
00BE	3A0000	E 155	lda scbpg
00C1	C3D700	C 156	jmp endt
		157 test7:	
00C4	3A0000	E 158	lda bnkoff
00C7	6F	159	mov l,a
00C8	1A	160	ldax d
00C9	94	161	sub h
00CA	DAD100	C 162	jc default

;if page = 0ffh

;then page = bios\$page

;else

;if page = 0feh

;then page = SCB\$page

;add 9ch to the offset(low byte)

;else

;if page = 0fdh

;then page = resbdos\$page

;else

;if page = 0fch

;then page = bnkhdos\$page

;else

;if page = 0fbh

;then page = scb\$page

;else

;if non\$banked and page >= reslen

;then do;

;page\$adr = page\$adr - 1;

;page = 9ah;

;page\$adr = page\$adr + 1;

;page = scb\$pg;

;end;

;else

;if page >= reslen

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

LOC	OBJ	SEQ	SOURCE STATEMENT
00CD	85	163	add l ;then page = page - reslen
00CE	C3D700	C 164	jmp endt
		165	default: ;else
00D1	3A0000	E 166	lda offset ;page = page + offset
00D4	67	167	mov h,a
00D5	1A	168	ldax d
00D6	84	169	add h
		170	endt:
00D7	12	171	stax d
00D8	E1	172	pop h
		173	pgrel4:
00D	13	174	inx d ;to next address
00DA	C33900	C 175	jmp pgrel0 ;for another byte to relocate
		176	
		177	ExitRelFix:
00DD	E1	178	pop h
00DE	210000	179	lxi h,0
00E1	7C	180	mov a,h
00E2	C9	181	ret
		182	
		183	errtn:
00E3	E1	184	pop h ;discard return address
00E4	21FFFF	185	lxi h,0ffffh
00E7	7C	186	mov a,h
00EB	C9	187	ret ;return with error condition
		188	;
		189	;
		190	;
0080		191	bitmap: ds 128 ;bit map buffer
0001		192	btmptp: ds 1 ;bit low (bitmap+128)
		193	
		194	end

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

PUBLIC SYMBOLS

BITMAP C 00E9 RELFIX C 0000

EXTERNAL SYMBOLS

BIOSPG E 0000	BNKOFF E 0000	BNKPG E 0000	BUFSIZ E 0000	FCBIN E 0000	MON1 E 0000	NONBNK E 0000
OFFSET E 0000	PRGSIZ E 0000	RESLEN E 0000	RESPG E 0000	SCBPG E 0000	SCTBFR E 0000	

USER SYMBOLS

BIOSPG E 0000	BITMAP C 00E9	BNKOFF E 0000	BNKPG E 0000	BTMPTP C 0169	BUFSIZ E 0000	DEFAULT C 00D1
ENDT C 00D7	ERRTN C 00E3	EXITRE C 00DD	FCBIN E 0000	MON1 E 0000	NOFILL C 0025	NONBNK E 0000
OFFSET E 00C	PGREL0 C 0039	PGREL2 C 0063	PGREL3 C 0067	PGREL4 C 00D9	PRGSIZ E 0000	RELFIX C 0000
RESLEN E 0000	RESPG E 0000	SCBPG E 0000	SCTBFR E 0000	TEST2 C 0079	TEST3 C 008B	TEST4 C 0095
TEST5 C 009F	TEST6 C 00A9	TEST7 C 00C4				

ASSEMBLY COMPLETE, NO ERRORS

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE GETSYSDEFAULTS
 OBJECT MODULE PLACED IN GETDEF.OBJ
 COMPILER INVOKED BY: PLM80 GETDEF.PLH DEBUG OPTIMIZE

```

1      $title('GENCPM Token File parser')
      get$sys$defaults:
      do;

      /*
      Copyright (C) 1982
      Digital Research
      P.O. Box 579
      Pacific Grove, CA 93950
      */

      /*
      Revised:
      20 Sept 82 by Bruce Skidmore
      */

2 1      declare true literally 'OFFH';
3 1      declare false literally '0';
4 1      declare forever literally 'while true';
5 1      declare boolean literally 'byte';
6 1      declare cr literally '0dh';
7 1      declare lf literally '0ah';
8 1      declare tab literally '09h';

      /*
      Data Structures
      */

9 1      declare data$fcbl (36) byte external;
10 1      declare quest (156) boolean external;
11 1      declare display boolean external;
12 1      declare symbol (8) byte;
13 1      declare lnbfr (14) byte external;
14 1      declare buffer (128) byte at (.memory);
15 1      declare symbtl (20) structure(
          token(8) byte,
          len      byte,
          flags    byte,
          qptr     byte,
          ptr      address) external;

16 1      mon1:
          procedure (func,info) external;
17 2          declare func byte;
18 2          declare info address;

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
19  2      end mon1;

20  1      mon2:
21  2          procedure (func,info) byte external;
22  2          declare func byte;
23  2          declare info address;
23  2      end mon2;

/*
   B D O S   P r o c e d u r e   &   F u n c t i o n   C a l l s
*/

24  1      system$reset:
25  2          procedure external;
25  2      end system$reset;

26  1      write$console:
27  2          procedure (char) external;
28  2          declare char byte;
28  2      end write$console;

29  1      print$console$buffer:
30  2          procedure (buffer$address) external;
31  2          declare buffer$address address;
31  2      end print$console$buffer;

32  1      open$file:
33  2          procedure (fcb$address) byte external;
34  2          declare fcb$address address;
35  2          declare fcb based fcb$address (1) byte;
35  2      end open$file;

36  1      close$file:
37  2          procedure (fcb$address) external;
38  2          declare fcb$address address;
38  2      end close$file;

39  1      set$DMA$address:
40  2          procedure (DMA$address) external;
41  2          declare DMA$address address;
41  2      end set$DMA$address;

42  1      crlf:
43  2          procedure external;
43  2      end crlf;

44  1      dsply$dec$adr:
45  2          procedure (val) external;
46  2          declare val address;
46  2      end dsply$dec$adr;

/*
   M a i n   G E T D E F   P r o c e d u r e
*/
47  1      getdef:
47  1          procedure public;
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

48 2    declare buffer$index byte;
49 2    declare index byte;
50 2    declare end$of$file byte;
51 2    declare line$count address;

52 2    err:
        procedure(term$code,msg$adr);
53 3        declare (term$code,save$display) byte;
54 3        declare msg$adr address;

55 3        save$display = display;
56 3        display = true;
57 3        call print$console$buffer(,('ERROR: '));
58 3        call print$console$buffer(msg$adr);
59 3        call print$console$buffer(,(' at line '));
60 3        call display$dec$adr(line$count);
61 3        if term$code then
62 3            call system$reset;
63 3            call crlf;
64 3            display = save$display;
65 3        end err;

66 2    inc$ptr:
        procedure;

67 3        if buffer$index = 127 then
68 3            do;
69 4            buffer$index = 0;
70 4            if mon2(20,,data$fcbl) <> 0 then
71 4                end$of$file = true;
72 4            end;
        else
73 3            buffer$index = buffer$index + 1;
74 3        end inc$ptr;

75 2    get$char:
        procedure byte;
76 3        declare char byte;

77 3        call inc$ptr;
78 3        char = buffer(buffer$index);
79 3        do while (char = ' ') or (char = tab) or (char = lf);
80 4            if char = lf then
81 4                line$count = line$count + 1;
82 4            call inc$ptr;
83 4            char = buffer(buffer$index);
84 4        end;
85 3        if (char >= 'a') and (char <= 'z') then
86 3            char = char and 0101$1111b; /* force upper case */
87 3        if char = lah then
88 3            end$of$file = true;
89 3        return char;
90 3    end get$char;

91 2    get$sym:
        procedure;
92 3        declare (i,sym$char) byte;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

93 3      declare got$sym boolean;

94 3      got$sym = false;
95 3      do while (not got$sym) and (not end$of$file);
96 4          do i = 0 to 7;
97 5              symbol(i) = ' ';
98 5          end;
99 4          sym$char = get$char;
100 4          i = 0;
101 4          do while (i < 8) and (sym$char <> '=') and
              (sym$char <> cr) and (not end$of$file);
102 5              symbol(i) = sym$char;
103 5              sym$char = get$char;
104 5              i = i + 1;
105 5          end;
106 4          do while (sym$char <> '=') and (sym$char <> cr) and (not end$of$file);
107 5              sym$char = get$char;
108 5          end;
109 4          if not end$of$file then
110 4              do;
111 5                  if (sym$char = '=') and (i > 0) then
112 5                      got$sym = true;
113 5                  else
114 5                      do;
115 6                          if (sym$char = '=') then
116 6                              call err(false,('Missing parameter variable$'));
117 6                          else
118 6                              if i < 0 then
119 6                                  call err(false,('Equals (=) delimiter missing$'));
120 6                                  do while (sym$char <> cr) and (not end$of$file);
121 6                                      sym$char = get$char;
122 6                                  end;
123 6                              end;
124 4                          end;
125 2          end get$sym;

126 2      get$val:
127 2      procedure;
128 3          declare (flags,i,val$char) byte;
129 3          declare val$adr address;
130 3          declare val based val$adr byte;
131 3          declare (base,inc,lbnfr$index) byte;

132 3          val$char = get$char;
133 3          i = 0;
134 3          do while (i < lbnfr(0)) and (val$char <> cr) and (not end$of$file);
135 4              lbnfr(i+2) = val$char;
136 4              i = i + 1;
137 4              lbnfr(1) = i;
138 4              val$char = get$char;
139 4          end;
140 3          do while (val$char <> cr) and (not end$of$file);
141 4              val$char = get$char;
142 4          end;
143 3          inc = 0;
144 3          lbnfr$index = 2;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

143 3      if i > 0 then
144 3          do;
145 4              val$adr = sytbl(index).ptr;
146 4              flags = sytbl(index).flags;
147 4              if (flags and 8) <> 0 then
148 4                  do;
149 5                      if (flags and 10h) <> 0 then
150 5                          inc = symbol(7) - 'A';
151 5                      else
152 5                          if (symbol(7) >= '0') and (symbol(7) <= '9') then
153 5                              inc = symbol(7) - '0';
154 5                          else
155 5                              inc = 10 + (symbol(7) - 'A');
156 5                          val$adr = val$adr + (inc * sytbl(index).len);
157 5                      end;
158 4              if lnbfr(lnbfr$index) = '?' then
159 4                  do;
160 5                      quest(inc+sytbl(index).qptr) = true;
161 5                      display = true;
162 5                      lnbfr$index = lnbfr$index + 1;
163 5                      lnbfr(1) = lnbfr(1) - 1;
164 5                  end;
165 4              if lnbfr(1) > 0 then
166 4                  do;
167 5                      if (flags and 1) <> 0 then
168 5                          do;
169 6                              if (lnbfr(lnbfr$index) >= 'A') and
170 6                                  (lnbfr(lnbfr$index) <= 'P') then
171 6                                  val = lnbfr(lnbfr$index) - 'A';
172 6                              else
173 6                                  call err(false,('Invalid drive ignored$'));
174 6                              end;
175 5                          else
176 5                              if (flags and 2) <> 0 then
177 5                                  do;
178 6                                      val = (lnbfr(lnbfr$index) - 'Y');
179 6                                      end;
180 5                                  else
181 5                                      do;
182 6                                          base = 16;
183 6                                          val = 0;
184 6                                          do i = 0 to lnbfr(1) - 1;
185 6                                              val$char = lnbfr(i+lnbfr$index);
186 6                                              if val$char = ',' then
187 6                                                  do;
188 7                                                      val$adr = val$adr + 1;
189 7                                                      val = 0;
190 7                                                      base = 16;
191 7                                                      end;
192 6                                                  else
193 6                                                      do;
194 7                                                          if val$char = '$' then
195 7                                                              base = 10;
196 7                                                          else
197 7                                                              do;
198 8                                                                  val$char = val$char - '0';
199 8                                                                  if (base = 16) and (val$char > 9) then

```

CP/M Plus V3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SFR. # *CP3-135*

```

192 9      do;
193 10      if val$char > 16 then
194 10      val$char = val$char - 7;
      else
195 10      val$char = 0ffh;
196 10      end;
197 9      if val$char < base then
198 9      val = val * base + val$char;
      else
199 9      call err(false,('Invalid character$'));
200 9      end;
201 8      end;
202 7      end;
203 6      end;
204 5      end;
205 4      end;
206 3      end get$val;

207 2      compare$sym:
      procedure byte;
208 3      declare (i,j) byte;
209 3      declare found boolean;

210 3      found = false;
211 3      i = 0;
212 3      do while ((i < 22) and (not found));
213 4      j = 0;
214 4      do while ((j < 7) and (syntbl(i).token(j) = symbol(j)));
215 5      j = j + 1;
216 5      end;
217 4      if j = 7 then
218 4      found = true;
      else
219 4      i = i + 1;
220 4      end;
221 3      if not found then
222 3      return 0ffh;
      else
223 3      return i;
224 3      end compare$sym;

225 2      line$count = 1;
226 2      call set$dma$address(.buffer);
227 2      buffer$index = 127;
228 2      end$of$file = false;
229 2      do while (not end$of$file);
230 3      call get$sym;
231 3      if not end$of$file then
232 3      do;
233 4      index = compare$sym;
234 4      if index <> 0ffh then
235 4      call get$val;
      else
236 4      call err(false,('Invalid parameter variable$'));
237 4      end;
238 3      end;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
239 2      end getdef;  
240 1      end get$sys$defaults;
```

MODULE INFORMATION:

```
CODE AREA SIZE   = 05D6H   1494D  
VARIABLE AREA SIZE = 0020H   32D  
MAXIMUM STACK SIZE = 0008H   8D  
338 LINES READ  
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

COPYRIGHT © 1932
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SETUPBUFFERS
 OBJECT MODULE PLACED IN SETBUF.OBJ
 COMPILER INVOKED BY: PLM80 SETBUF.PLX DEBUG OPTIMIZE

```
1      $title ('GENCPM - Buffer allocation module')
      setup$buffers;
      do;
```

```
/*
  Copyright (C) 1982
  Digital Research
  P.O. Box 579
  Pacific Grove, CA 93950
  */
```

```
/*
  Revised:
    09 Dec 82 by Bruce Skidmore
  */
```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
2 1      declare true literally 'OFFH';
3 1      declare false literally '0';
4 1      declare forever literally 'while true';
5 1      declare boolean literally 'byte';
6 1      declare cr literally '0dh';
7 1      declare lf literally '0ah';
```

```
/*
  Data Structures
  */
```

```
8 1      declare query boolean external;
9 1      declare quest(155) boolean external;

10 1     declare offset byte external;
11 1     declare prgsiz address external;
12 1     declare bufsiz address external;
13 1     declare codsiz address external;
14 1     declare bios$pg byte external;
15 1     declare scb$pg byte external;
16 1     declare res$pg byte external;
17 1     declare bnk$pg byte external;
18 1     declare bnk$off byte external;
19 1     declare res$len byte external;
20 1     declare non$bnk byte external;

21 1     declare dma address external;
22 1     declare lnbfr (14) byte external;

23 1     declare bios$atts(3) address external;
24 1     declare res$atts(3) address external;
25 1     declare bnk$atts(3) address external;

26 1     declare res$bios$len byte external;
27 1     declare res$base byte external;
```

```

28 1      declare pg$dif byte external;
29 1      declare xmove$implemented boolean external;

30 1      declare mem$top byte external;
31 1      declare common$len byte external;
32 1      declare bnk$top byte external;
33 1      declare banked$len byte external;
34 1      declare sys$entry address external;
35 1      declare bnk$swt boolean external;

36 1      declare drvtbl$adr address external;
37 1      declare drvtbl based drvtbl$adr (16) address;

38 1      declare dph$adr address external;
39 1      declare dph based dph$adr structure (
        xlt      address,
        scratch1(4) address,
        scratch2  byte,
        mf        byte,
        dpb        address,
        csv        address,
        alv        address,
        dirbcb     address,
        dtabcb     address,
        hash       address,
        hbank      byte);

40 1      declare dpb$adr address external;
41 1      declare dpb based dpb$adr structure (
        spt      address,
        bsh      byte,
        blm      byte,
        exm      byte,
        dsm      address,
        drn      address,
        al0      byte,
        al1      byte,
        cks      address,
        off      address,
        psh      byte,
        phm      byte);

42 1      declare mem$tbl (17) structure(
        base      byte,
        len       byte,
        bank      byte,
        attr      address) external;

43 1      declare num$seg byte external;

44 1      declare record(16) structure(
        size      address,
        attr      byte,
        altbnks   byte,
        no$dirrecs byte,
        no$dtarecs byte,
        ovlydir$dr byte,

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

        ovlydta$dr byte,
        dir$resp  byte,
        dta$resp  byte) external;

45  1      declare hash$data(16) address external;
46  1      declare hash$space address external;
47  1      declare hash(16) boolean external;
48  1      declare alloc(16) address external;
49  1      declare alloc$space address external;
50  1      declare chk(16) address external;
51  1      declare chk$space address external;

/*
      B D O S  P r o c e d u r e   &   F u n c t i o n   C a l l s
*/

52  1      system$reset:
        procedure external;
53  2      end system$reset;

54  1      write$console:
        procedure (char) external;
55  2      declare char byte;
56  2      end write$console;

57  1      print$console$buffer:
        procedure (buffer$address) external;
58  2      declare buffer$address address;
59  2      end print$console$buffer;

60  1      read$console$buffer:
        procedure (buffer$address) external;
61  2      declare buffer$address address;
62  2      declare buf based buffer$address (1) byte;
63  2      end read$console$buffer;

/*
      L o c a l   P r o c e d u r e s
*/

64  1      shift$left:
        procedure (pattern, count) address external;
65  2      declare count byte;
66  2      declare pattern address;
67  2      end shift$left;

68  1      crlf:
        procedure external;
69  2      end crlf;

70  1      error:
        procedure (term$code,err$type,err$msg$adr) external;
71  2      declare (term$code,err$type) byte;
72  2      declare err$msg$adr address;
73  2      end error;

74  1      upper:

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

      procedure (b) byte external;
75  2      declare b byte;
76  2      end upper;

77  1      valid$drive:
      procedure(drv) boolean external;
78  2      declare drv byte;
79  2      end valid$drive;

80  1      get$response:
      procedure (val$adr) external;
81  2      declare val$adr address;
82  2      end get$response;

83  1      dsply$hex$adr:
      procedure (val) external;
84  2      declare val address;
85  2      end dsply$hex$adr;

86  1      get$param:
      procedure (string$adr,val$adr,pbase) external;
87  2      declare (string$adr,val$adr) address;
88  2      declare pbase byte;
89  2      end get$param;

90  1      get$seg:
      procedure(type,record$size) byte external;
91  2      declare type byte;
92  2      declare record$size address;
93  2      end get$seg;

94  1      setbuf:
      procedure public;
95  2      declare (i,j,k,ii,save,data$cnt,temp) byte;
96  2      declare (first$dir,first$dta,first$drive,other$banks) boolean;
97  2      declare (ok,perm$media,printed) boolean;
98  2      declare (link$cnt,seq$no,dir$data$field) byte;
99  2      declare mem$save$tbl(17) address;
100 2      declare sav$mem$len (17) byte;
101 2      declare (rec$siz,drives,save$dph$adr,save$bcb$adr) address;
102 2      declare (bcb$cnt,bcb$buf$siz,buff$space) address;
103 2      declare (data$adr,l$buf$adr,act$buf$adr,l$head$adr) address;
104 2      declare (max$dir$buf,max$attr,tpa,defined$drives) address;
105 2      declare l$head based l$head$adr address;

106 2      declare bcb$len byte;
107 2      declare bcb$buf$cnt address;
108 2      declare bcb$buf$ptr byte;

109 2      declare bcb$adr address;
110 2      declare bcb based bcb$adr structure(
          drv      byte,
          rec$no(3) byte,
          wflag     byte,
          seq$no    byte,
          track     address,
          sector    address,

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

buff$adr  address,
bank      byte,
link      address);

111  2      declare psect (16) structure(
           size  address,
           drives address);

112  2      disp$space:
           procedure;
113  3          declare (seg0,obanks) address;
114  3          declare ii byte;

115  3          seg0, obanks = 0;
116  3          do ii = 1 to num$seg;
117  4              if mem$tbl(ii).bank = 0 then
118  4                  seg0 = seg0 + shr(mem$tbl(ii).attr,8);
           else
119  4                  obanks = obanks + shr(mem$tbl(ii).attr,8);
120  4          end;
121  3          call print$console$buffer(.(lf,cr,'      ',
                                           'Available space in 256 byte pages:',
                                           lf,cr,'      ','$'));

122  3          call print$console$buffer(.( 'TPA =$'));
123  3          call dsply$hex$adr(shr(tpa,8));
124  3          if bnk$swt then
125  3              do;
126  4                  call print$console$buffer(.( ' , Bank 0 =$'));
127  4                  call dsply$hex$adr(seg0);
128  4                  if xmove$implemented then
129  4                      do;
130  5                          call print$console$buffer(.( ' , Other banks =$'));
131  5                          call dsply$hex$adr(obanks);
132  5                      end;
133  4                  if (obanks < 0) or (seg0 < 0) then
134  4                      other$banks = true;
           else
135  4                      other$banks = false;
136  4                  end;
137  3                  call crlf;
138  3                  call crlf;
139  3          end disp$space;

140  2      get$space:
           procedure(index,param) byte;
141  3          declare (index,param,count,ii) byte;
142  3          declare (i,j,k,save) byte;
143  3          declare seg$no byte;
144  3          declare rsize address;

145  3          rsize = record(index).size;
146  3          if not bnk$swt then
147  3              do;
148  4                  tpa = tpa - (rsize + bcb$len);
149  4              end;
           else

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

150 3      do;
151 4      bcb$buf$ptr = bcb$buf$ptr - 2;
152 4      if parm = 1 then /* directory records */
153 4      do;
154 5          count = record(index).no$dirrecs;
155 5          do ii = 1 to count;
156 6              if bcb$buf$ptr < bcb$len then
157 6              do;
158 7                  bcb$buf$ptr = 0ffh;
159 7                  bcb$buf$cnt = bcb$buf$cnt + 1;
160 7                  j = 0ffh;
161 7                  save = 0;
162 7                  do i = 1 to num$seg;
163 8                      k = mem$tbl(i).len + mem$tbl(i).base;
164 8                      if (mem$tbl(i).bank = 0) and k > save then
165 8                      do;
166 9                          j = i;
167 9                          save = k; /* pre allocate space for BCBs */
168 9                      end;
169 8                  end;
170 7                  mem$tbl(j).attr = mem$tbl(j).attr - 100h;
171 7              end;
172 6          else
173 6              bcb$buf$ptr = bcb$buf$ptr - bcb$len;
174 6          if (seg$no := get$seg(1, rsize)) = 0ffh then
175 7          do;
176 7              call error(false, 0,
177 7                  (('Unable to allocate Dir deblocking ',
178 7                  'buffer space.$'));
179 7              return false;
180 5          end;
181 4          else
182 5              mem$tbl(seg$no).attr = mem$tbl(seg$no).attr - rsize;
183 5          end;
184 5      end;
185 5      /* data records */
186 5      do;
187 6          count = record(index).no$dtarecs;
188 6          do ii = 1 to count;
189 6          if bcb$buf$ptr < bcb$len then
190 6          do;
191 7              bcb$buf$ptr = 0ffh;
192 7              bcb$buf$cnt = bcb$buf$cnt + 1;
193 7              j = 0ffh;
194 7              save = 0;
195 7              do i = 1 to num$seg;
196 8                  k = mem$tbl(i).len + mem$tbl(i).base;
197 8                  if (mem$tbl(i).bank = 0) and k > save then
198 8                  do;
199 9                      j = i;
200 9                      save = k; /* pre allocate space for BCBs */
201 9                  end;
202 8              end;
203 7              mem$tbl(j).attr = mem$tbl(j).attr - 100h;
204 7          end;
205 6          else

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

200 6          bcb$buf$ptr = bcb$buf$ptr - bcb$len;
201 6          if not record(index).altbnks then
202 6              do;
203 7              tpa = tpa - rsize;
204 7              if shr(tpa,8) < bnk$top then
205 7                  do;
206 8                      call error(false,0,
                              .('Unable to allocate Data ',
                                'deblocking buffer space.$'));
207 8                      return false;
208 8                  end;
209 7              end;
          else
210 6              do;
211 7                  if (seg$no := get$seg(2,rsize)) = 0ffh then
212 7                      do;
213 8                          call error(false,0,
                              .('Unable to allocate Data deblocking ',
                                'buffer space.$'));
214 8                          return false;
215 8                      end;
          else
216 7              mem$tbl(seg$no).attr = mem$tbl(seg$no).attr - rsize;
217 7              end;
218 6          end;
219 5          end;
220 4          end;
221 3          return true;
222 3      end get$space;

223 2      drive$not$defined;
          procedure(val) boolean;
224 3          declare val byte;
225 3          if (defined$drives and shift$left(double(1),val)) <> 0 then
226 3              return false;
227 3          call error(false,0,('Drive specified has not ',
                                'been defined. $'));
228 3          return true;
229 3      end drive$not$defined;

230 2      do i = 0 to 15;
231 3          if drvtbl(i) <> 0 then
232 3              do;
233 4              dph$adr = drvtbl(i) + .memory;
234 4              if (dph.dirbcb = 0ffffh) or (dph.dtabcb = 0ffffh) then
235 4                  do;
236 5                  dpb$adr = dph.dpb + .memory;
237 5                  record(i).size = shift$left(double(128),dpb,psh);
238 5                  record(i).attr = 0;
239 5                  end;
240 4              if dph.dirbcb = 0ffffh then
241 4                  do;
242 5                  call error(true,0,('0FFFFH is an invalid value in the',
                                        'cr,lf',
                                        'DPH directory BCB address field.$'));
243 5                  end;

```

COPYRIGHT © 1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```
244 4      end;
245 3      end;

246 2      do j = 0 to 15;
247 3          rec$siz = 0;
248 3          do i = 0 to 15;
249 4              if (record(i).size > rec$siz) and (record(i).attr = 0) then
250 4                  rec$siz = record(i).size;
251 4              end;
252 3              psect(j).size = rec$siz;
253 3              psect(j).drives = 0;
254 3              do i = 0 to 15;
255 4                  if (record(i).size = rec$siz) and (rec$siz <> 0) then
256 4                      do;
257 5                          psect(j).drives = psect(j).drives or shift$left(double(1),i);
258 5                          record(i).attr = 0ffh;
259 5                      end;
260 4              end;
261 3      end;

262 2      do i = 0 to 16;
263 3          mem$sav$tbl(i) = mem$tbl(i).attr;
264 3          sav$mem$len(i) = mem$tbl(i).len;
265 3      end;

266 2      if bnk$swt then
267 2          bcb$len = 15;
268 2      else
268 2          bcb$len = 12;

269 2      ok = false;
270 2      do while not ok;

271 3          bcb$buf$ptr = bcb$len - 1;
272 3          bcb$buf$cnt = 0;
273 3          defined$drives = 0;
274 3          data$cnt = 0;
275 3          first$drive = true;
276 3          tpa = shl(double(res$pg),8) - hash$space;
277 3          if not bnk$swt then
278 3              tpa = tpa - alloc$space - chk$space;

279 3          do i = 0 to 15;
280 4              mem$tbl(i+1).attr = mem$sav$tbl(i+1);
281 4              record(i).attr = 0;
282 4          end;

283 3          printed = false;
284 3          first$dir, first$dta = true;

285 3          ii = 0;
286 3          drives = psect(ii).drives;
287 3          do while (psect(ii).size <> 0) and (ii < 16);

288 4              if not printed then
```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

289 4      do;
290 5          call print$console$buffer(.(lf,cr,'Setting up ',
                                         'Blocking/Deblocking buffers:',
                                         lf,cr,'$'));

291 5          printed = true;
292 5      end;

293 4      call print$console$buffer(
          .(cr,lf,'The physical record size is$'));
294 4      call dsply$hex$adr(psect(ii).size);
295 4      call print$console$buffer(.((' ',lf,cr,'$'));

296 4      i = 0;
297 4      do while((drives and 1) <> 1) and (i < 16);
298 5          drives = shr(drives,1);
299 5          i = i + 1;
300 5      end;
301 4      drives = shr(drives,1);
302 4      do while (i <> 16);
303 5          dph$adr = drvtbl(i) + .memory;
304 5          dpb$adr = dph.dpb + .memory;
305 5          max$dir$buf = shr(dpb.drm + 4,2);
306 5          if (dpb.cks = 08000h) then
307 5              perm$media = true;
          else
308 5              perm$media = false;
309 5          if dph.dirbcb = 0ffffh then
310 5              do;
311 6                  call disp$space;
312 6                  record(i).attr = record(i).attr or 1;
313 6                  if not bnk$swt then
314 6                      do;
315 7                          if first$dir then
316 7                              do;
317 8                              first$dir = false;
318 8                              record(i).no$dirrecs = 1;
319 8                              call print$console$buffer(
                                  .(' ',
                                    '*** Directory buffer required ***',
                                    cr,lf,' ',
                                    '*** and allocated for drive $'));
320 8                              call write$console('A'+i);
321 8                              call print$console$buffer(.((' ' ***',cr,lf,'$'));
322 8                          end;
          else
323 7              do;
324 8                  query = quest(123 + i);
325 8                  call print$console$buffer(
                      .(' ',
                        'Overlay Directory buffer for drive $'));
326 8                  call write$console('A'+i);
327 8                  call print$console$buffer(.((' ' $'));
328 8                  call get$response(.record(i).dir$resp);
329 8                  call crlf;
330 8                  if not record(i).dir$resp then
331 8                      do;
332 9                      record(i).no$dirrecs = 1;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

333 9          end;
          else
334 8          record(i).no$dirrecs = 0;
335 8          end;
336 7          end;
        else
337 6          do;
338 7          query = quest(59 + i);
339 7          call print$console$buffer(
              .('
              'Number of directory buffers for drive $'));
340 7          call write$console('A'ti);
341 7          call get$param((': $'),record(i).no$dirrecs,10);
342 7          if first$dir then
343 7          do;
344 8          first$dir = false;
345 8          do while (record(i).no$dirrecs = 0);
346 9          call error(false,0,
              .('Minumum number of buffers is 1. $'));
347 9          call print$console$buffer(
              .('
              'Number of directory buffers for drive $'));
348 9          call write$console('A'ti);
349 9          call get$param((': $'),record(i).no$dirrecs,10);
350 9          end;
351 8          end;
352 7          end;

353 6          if record(i).no$dirrecs > max$dir$buf then
354 6          do;
355 7          call print$console$buffer(
              .(cr,lf,'*** Maximum number of directory buffers ***',
              cr,lf,'*** for the current drive is$'));
356 7          call dsply$hex$adr(max$dir$buf);
357 7          call print$console$buffer(('. ***',
              cr,lf,'*** Number of directory buffers reduced ***',
              cr,lf,'*** accordingly. ***',
              cr,lf,'$'));
358 7          record(i).no$dirrecs = max$dir$buf;
359 7          end;

360 6          if record(i).no$dirrecs = 0 then
361 6          do;
362 7          query = quest(91 + i);
363 7          err4:
          call print$console$buffer(
              .('
              'Share buffer(s) with which drive ($'));
364 7          call write$console('A'trecord(i).ovlydir$dr);
365 7          call print$console$buffer((': ? $'));
366 7          call read$console$buffer(.lnbfr);
367 7          if lnbfr(1) <> 0 then
368 7          do;
369 8          temp = upper(lnbfr(2))- 'A';
370 8          if not valid$drive(temp) then
371 8          goto err4;
372 8          if drive$not$defined(temp) then

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

373 8          goto err4;
374 8          record(i).ovlydir$dr = temp;
375 8          end;
376 7          call crlf;
377 7          end;
          else
378 6              if (get$space(i,1) = 0) then
379 6                  goto notok;
          end;

381 5          if (dph.dtabcb = 0ffffh) then
382 5              do;
383 6                  dir$data$field = 0;
384 6                  if record(i).size = 80h then
385 6                      do;
386 7                          dph.dtabcb = 0ffffh;
387 7                          record(i).no$dta$recs = 0;
388 7                          end;
          else
389 6              do;
390 7                  call disp$space;
391 7                  record(i).attr = record(i).attr or 2;
392 7                  if not bnk$swt then
393 7                      do;
394 8                          if data$cnt < 2 then
395 8                              do;
396 9                                  if not perm$media then
397 9                                      do;
398 10                                         data$cnt = 2;
399 10                                         dir$data$field = 0ffh;
400 10                                         record(i).no$dta$recs = 1;
401 10                                         call print$console$buffer(
                                         .('          ',
                                         '*** Data buffer required and ***',
                                         cr,lf,'          ',
                                         '*** allocated for drive $'));
402 10                                         call write$console('A'+i);
403 10                                         call print$console$buffer(.('!   ***',cr,lf,'$'));
404 10                                         end;
          else
405 9              do;
406 10                  if first$dta then
407 10                      dir$data$field = 10h;
408 10                  end;
409 9              end;

410 8              first$dta = false;
411 8              if dir$data$field < 0ffh then
412 8                  do;
413 9                      query = quest(139 + i);
414 9                      call print$console$buffer(
                      .('          ',
                      'Overlay Data buffer for drive $'));
415 9                      call write$console('A'+i);
416 9                      call print$console$buffer(.('!   $'));
417 9                      call get$response(.record(i).dta$resp);
418 9                      call crlf;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

419 9          if record(i).dta$resp then /* Y */
420 9              record(i).no$dta$recs = 0;
              else
421 9                  do;
422 10                     record(i).no$dta$recs = 1;
423 10                     data$cnt = 2;
424 10                     end;
425 9              end;
426 8          end;
          else
427 7              do;
428 8                  query = quest(75 + i);
429 8                  call print$console$buffer(
                      .('
                        'Number of data buffers for drive $')));
430 8                  call write$console('A'ti);
431 8                  call get$param(.((' $'),,record(i).no$dta$recs,10);
432 8                  record(i).attr = record(i).attr or 2;
433 8                  if first$dta then
434 8                      do;
435 9                          first$dta = false;
436 9                          do while (record(i).no$dta$recs = 0);
437 10                             call error(false,0,
                                .('Minimum number of buffers is 1. $'));
438 10                             call print$console$buffer(
                                .('
                                  'Number of data buffers for drive $'));
439 10                             call write$console('A'ti);
440 10                             call get$param(.((' $'),,record(i).no$dta$recs,10);
441 10                             end;
442 9                          end;
443 8                      end;

444 7          if record(i).no$dta$recs = 0 then
445 7              do;
446 8                  if first$drive then
447 8                      do;
448 9                          first$drive = false;
449 9                          record(i).ovlydta$dr = i;
450 9                      end;
451 8                  else
452 9                      do;
453 9                          query = quest(107 + i);
err5:
                          call print$console$buffer(
                              .('
                                'Share buffer(s) with which drive ($'));
454 9                          call write$console('A'+record(i).ovlydta$dr);
455 9                          call print$console$buffer(.((' ? $'));
456 9                          call read$console$buffer(.lnbfr);
457 9                          if lnbfr(1) <> 0 then
458 9                              do;
459 10                                 temp = upper(lnbfr(2))- 'A';
460 10                                 if not valid$drive(temp) then
461 10                                     goto err5;
462 10                                 if drive$not$defined(temp) then
463 10                                     goto err5;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

464 10          record(i).ovlydta$dr = temp;
465 10          end;
466 9           call crlf;
467 9           end;
468 8           record(i).ovlydta$dr = record(i).ovlydta$dr
              or dir$data$field;
469 8           end;
              else
470 7             do;
471 8               if (other$banks and xmove$implemented) then
472 8                 do;
473 9                   query = quest(43 + i);
474 9                   call print$console$buffer(
                      .('
755 9                   'Allocate buffers outside of Common $'));
756 9                   call get$response(.record(i).altbnks);
757 9                   call crlf;
758 9                   end;
478 8                   if (get$space(i,2) = 0) then
479 8                     goto notok;
480 8                   end;
481 7                 end;
482 6             end;
483 5             first$drive = false;
484 5             defined$drives = defined$drives or shift$left(double(1),i);
485 5             i = i + 1;
486 5             do while((drives and 1) <> 1) and (i < 16);
487 6               drives = shr(drives,1);
488 6               i = i + 1;
489 6             end;
490 5             drives = shr(drives,1);
491 5             end;
492 4             ii = ii + 1;
493 4             drives = psect(ii).drives;
494 4             end;

495 3             query = false;
496 3             ok = true;
497 3             if printed then
498 3               do;
499 4                 call disp$space;
500 4                 call print$console$buffer(
                      (cr,lf,'Accept new buffer definitions $'));
501 4                 call get$response(.ok);
502 4                 call crlf;
503 4                 end;

504 3             notok: /*** start over here upon error ***/
end; /*** do while not ok ***/

/* calculate BCB requirements */

505 2             bcb$cnt,link$cnt,buff$space = 0;
506 2             do i = 0 to 15;
507 3               if record(i).attr < 0 then
508 3                 do;
509 4                   if (record(i).no$dirrecs < 0) and

```

COPYRIGHT © 1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

510 4      ((record(i),attr and 1) = 1) then
511 5      do;
512 5          if bnk$swt then
513 5              link$cnt = link$cnt + 1;
514 5          else
515 5              buff$space = buff$space +
516 5                  (record(i).size * record(i).no$dirrecs);
517 5              bcb$cnt = bcb$cnt + record(i).no$dirrecs;
518 5          end;
519 5          if (record(i).no$dtarecs <> 0) and
520 5              ((record(i),attr and 2) = 2) then
521 5              do;
522 5                  if bnk$swt then
523 5                      do;
524 5                          link$cnt = link$cnt + 1;
525 5                          if not record(i).altbnks then
526 5                              buff$space = buff$space + record(i).size
527 5                                  * record(i).no$dtarecs;
528 5                      end;
529 5                  else
530 5                      buff$space = buff$space +
531 5                          (record(i).size * record(i).no$dtarecs);
532 5                      bcb$cnt = bcb$cnt + record(i).no$dtarecs;
533 5                  end;
534 5              end;
535 5          end;
536 5          bcb$buf$siz = bcb$cnt * bcb$len + link$cnt * 2;

```

/** allocate deblocking buffers **/

```

530 2      if not bnk$swt then      /* for non-banked system */
531 2      do;
532 3          bcb$adr = bios$atts(0) + .memory;
533 3          pg$dif = bios$pg - (mem$top - high(bios$atts(0) + bcb$buf$siz +
534 3              buff$space + hash$space +
535 3              chk$space + alloc$space + 255));
536 3          bios$pg = bios$pg - pg$dif;
537 3          res$pg = res$pg - pg$dif;
538 3          scb$pg = scb$pg - pg$dif;
539 3          res$bios$len = high(bios$atts(0) + bcb$buf$siz + 255);
540 3          mem$top = bios$pg + res$bios$len;
541 3          dma = bios$atts(0) + bcb$buf$siz + .memory;
542 3          act$buf$adr = shl(double(bios$pg),8) + bios$atts(0);
543 3          data$adr = act$buf$adr + bcb$buf$siz;

```

/** zero memory for the BCB buffers **/

```

542 3      max$attr = prgsiz;
543 3      do while(.memory(max$attr) < dma);
544 3          memory(max$attr) = 0;
545 3          max$attr = max$attr + 1;
546 3      end;

547 3      ii = 0;
548 3      drives = psect(ii).drives;
549 3      do while (psect(ii).size <> 0) and (ii < 16);
550 3          i = 0;

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

551 4      do while((drives and 1) <> 1) and (i < 16);
552 5          drives = shr(drives,1);
553 5          i = i + 1;
554 5      end;
555 4      drives = shr(drives,1);
556 4      do while(i <> 16);
557 5          dph$adr = drvtbl(i) + .memory;
558 5          if (record(i).attr and 1) = 1 then
559 5              do;
560 6                  if record(i).no$dirrecs <> 0 then
561 6                      do;
562 7                          bcb.driv = 0ffh;
563 7                          bcb.buff$adr = data$adr;
564 7                          data$adr = data$adr + record(i).size;
565 7                          dph.dirbcb = act$buf$adr;
566 7                          act$buf$adr = act$buf$adr + bcb$len;
567 7                          bcb$adr = bcb$adr + bcb$len;
568 7                      end;
569 6                  else
570 7                      do;
571 7                          save$dph$adr = dph$adr;
572 7                          dph$adr = drvtbl(record(i).ovlydir$dr) + .memory;
573 7                          save$bcb$adr = dph.dirbcb;
574 7                          dph$adr = save$dph$adr;
575 7                          dph.dirbcb = save$bcb$adr;
576 6                      end;
577 5          end;
578 5          if (record(i).attr and 2) = 2 then
579 6              do;
580 6                  if record(i).no$dtarecs <> 0 then
581 7                      do;
582 7                          bcb.driv = 0ffh;
583 7                          bcb.buff$adr = data$adr;
584 7                          data$adr = data$adr + record(i).size;
585 7                          dph.dtabcb = act$buf$adr;
586 7                          act$buf$adr = act$buf$adr + bcb$len;
587 7                          bcb$adr = bcb$adr + bcb$len;
588 6                      end;
589 6                  else
590 7                      do;
591 7                          save$dph$adr = dph$adr;
592 7                          dph$adr = drvtbl((record(i).ovlydta$dr) and 0fh) + .memory;
593 7                          if (record(i).ovlydta$dr and 10h) <> 0 then
594 7                              save$bcb$adr = dph.dirbcb;
595 7                          else
596 7                              save$bcb$adr = dph.dtabcb;
597 7                          dph$adr = save$dph$adr;
598 7                          dph.dtabcb = save$bcb$adr;
599 6                      end;
600 5              end;
601 5              i = i + 1;
602 5          do while((drives and 1) <> 1) and (i < 16);
603 5              drives = shr(drives,1);
604 5              i = i + 1;
605 5          end;
606 4          drives = shr(drives,1);

```

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

604 5          end;
605 4          ii = ii + 1;
606 4          drives = psect(ii),drives;
607 4      end;

608 3      do i = 0 to 15;          /* allocate hash for non-bank system */
609 4          if hash$data(i) <> 0 then
610 5              do;
611 6                  dph$adr = drvtbl(i) + .memory;
612 6                  dph,hash = data$adr;
613 6                  data$adr = data$adr + hash$data(i);
614 6              end;
615 4      end;

616 3      do i = 0 to 15;          /* allocate allocation vectors and */
                                /* checksub vectors for non-bank system */
617 4          dph$adr = drvtbl(i) + .memory;
618 4          if alloc(i) <> 0 then
619 5              do;
620 6                  dph,alv = data$adr;
621 6                  data$adr = data$adr + alloc(i);
622 6              end;
623 4          if chk(i) <> 0 then
624 5              do;
625 6                  dph,csv = data$adr;
626 6                  data$adr = data$adr + chk(i);
627 6              end;
628 4          end;

629 3      end;
else          /* allocate deblocking buffers for banked system */
630 2      do;
                                /* restore memory table */
631 3          do i = 0 to 16;
632 4              mem$tbl(i).attr = mem$sav$tbl(i);
633 4          end;

634 3          bcb$buf$siz = bcb$buf$siz + chk$space + alloc$space;
635 3          if (bios$atts(2) = 0) or (bios$atts(0) = bios$atts(2)) then
636 3              pg$dif = bios$pg - (mem$top - high(bios$atts(0) + buff$space + 255));
          else
637 3              pg$dif = bios$pg - (mem$top - high(bios$atts(2) + buff$space + 255));
638 3          bios$pg = bios$pg - pg$dif;
639 3          res$pg = res$pg - pg$dif;
640 3          scb$pg = scb$pg - pg$dif;
641 3          mem$top = mem$top - pg$dif;
642 3          max$attr = bios$atts(0) - (bios$atts(2) + low(256-low(bios$atts(2))));
643 3          if (bios$atts(2) = 0) or (bios$atts(0) = bios$atts(2)) then
644 4              do;
645 5                  bnk$off = bnk$top - high(bcb$buf$siz + 255);
646 5                  l$head$adr = bios$atts(0) + low(256-low(bios$atts(0))) + .memory;
647 5                  l$buf$adr = shl(double(bnk$off),8);
648 5                  data$adr = shl(double(bios$pg),8) + bios$atts(0);
649 5              end;
          else
650 3              do;
651 4                  bnk$off = bnk$top - high(max$attr + bcb$buf$siz + 255);

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

652 4      l$head$adr = bios$atts(0) + .memory;
653 4      l$buf$adr = shl(double(bnk$off),8) + max$attr;
654 4      data$adr = shl(double(bios$pg),8) + bios$atts(2);
655 4      end;
656 3      dma = l$head$adr + bcb$buf$size;
657 3      bnk$pg = bnk$off - high(bnk$atts(0) + 255);
658 3      bcb$adr = l$head$adr + link$cnt * 2;
659 3      act$buf$adr = l$buf$adr + link$cnt * 2;
660 3      bios$atts(0) = dma - .memory;

      /*** zero memory for the BCB buffers ***/

661 3      max$attr = prgsiz;
662 3      do while(.memory(max$attr) < dma);
663 4          memory(max$attr) = 0;
664 4          max$attr = max$attr + 1;
665 4      end;

      /*** allocate memory table space for BCB's ***/

666 3      j = 0ffh;
667 3      save = 0;
668 3      do i = 1 to num$seg;
669 4          k = mem$tbl(i).len + mem$tbl(i).base;
670 4          if (mem$tbl(i).bank = 0) and (k > save) then
671 4              do;
672 5                  j = i;
673 5                  save = k;
674 5              end;
675 4      end;
676 3      mem$tbl(0).base = bnk$pg;
677 3      mem$tbl(0).len = bnk$top - bnk$pg;
678 3      if j < 0ffh then
679 3          do;
680 4              if (mem$tbl(j).len + mem$tbl(j).base) > bnk$pg then
681 4                  do;
682 5                      max$attr = shl(double(mem$tbl(j).len),8) - mem$tbl(j).attr;
683 5                      mem$tbl(j).len = bnk$pg - mem$tbl(j).base;
684 5                      mem$tbl(j).attr = shl(double(mem$tbl(j).len),8) - max$attr;
685 5                  end;
686 4              end;

      /*** allocate directory buffers for banked system ***/

687 3      ii = 0;
688 3      drives = psect(ii).drives;
689 3      do while (psect(ii).size <> 0) and (ii < 16);
690 4          i = 0;
691 4          do while ((drives and 1) <> 1) and (i < 16);
692 5              drives = shr(drives,1);
693 5              i = i + 1;
694 5          end;
695 4          drives = shr(drives,1);
696 4          do while (i < 16);
697 5              dph$adr = drvtbl(i) + .memory;
698 5              if (record(i).attr and 1) = 1 then
699 5                  do;
700 6                      if record(i).no$dirrecs = 0 then

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

701 6      do;
702 7          save$dph$adr = dph$adr;
703 7          dph$adr = drvtbl(record(i),ovlydir$dr) + .memory;
704 7          save$bcb$adr = dph.dirbcb;
705 7          dph$adr = save$dph$adr;
706 7          dph.dirbcb = save$bcb$adr;
707 7      end;
      else
708 6      do;
709 7          l$head = act$buf$adr; /** set up list head **/
710 7          dph.dirbcb = l$buf$adr;
711 7          l$buf$adr = l$buf$adr + 2;
712 7          l$head$adr = l$head$adr + 2;
              /** create bcbs **/
713 7          do j = 1 to record(i).no$dirrecs;
714 8              seg$no = get$seg(1,record(i).size);
715 8              bcb.driv = 0ffh;
716 8              bcb.buff$adr = shl(double(mem$tbl(seg$no).base),8) +
                          (shl(double(mem$tbl(seg$no).len),8) -
                          mem$tbl(seg$no).attr);
717 8              bcb.bank = 0;
718 8              mem$tbl(seg$no).attr = mem$tbl(seg$no).attr -
                          record(i).size;
719 8              act$buf$adr = act$buf$adr + bcb$len;
720 8              bcb.link = act$buf$adr;
721 8              bcb$adr = bcb$adr + bcb$len;
722 8          end;
723 7          bcb$adr = bcb$adr - bcb$len;
724 7          bcb.link = 0;
725 7          bcb$adr = bcb$adr + bcb$len;
726 7      end;
727 6      end;

```

/** allocate data deblocking buffers for banked system **/

```

728 5      if (record(i).attr and 2) = 2 then
729 5          do;
730 6              if record(i).no$dtarecs = 0 then
731 6                  do;
732 7                      save$dph$adr = dph$adr;
733 7                      dph$adr = drvtbl(record(i),ovlydta$dr) + .memory;
734 7                      save$bcb$adr = dph.dtabcb;
735 7                      dph$adr = save$dph$adr;
736 7                      dph.dtabcb = save$bcb$adr;
737 7                  end;
              else
738 6                  do;
739 7                      l$head = act$buf$adr; /** set up list head **/
740 7                      dph.dtabcb = l$buf$adr;
741 7                      l$buf$adr = l$buf$adr + 2;
742 7                      l$head$adr = l$head$adr + 2;
                          /** create bcbs **/
743 7                      do j = 1 to record(i).no$dtarecs;
744 8                          if record(i).altbnks then
745 8                              do;
746 9                                  seg$no = get$seg(2,record(i).size);
747 9                                  bcb.driv = 0ffh;

```

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```

748 9          bcb.buff$adr = shl(double(mem$tbl(seg$no).base),8)
          + (shl(double(mem$tbl(seg$no).len),8)
          - mem$tbl(seg$no).attr);
749 9          bcb.bank = mem$tbl(seg$no).bank;
750 9          mem$tbl(seg$no).attr = mem$tbl(seg$no).attr -
          record(i).size;
751 9          end;
          else
752 8          do;
753 9          bcb.driv = 0ffh;
754 9          bcb.buff$adr = data$adr;
755 9          data$adr = data$adr + record(i).size;
756 9          bcb.bank = 0;
757 9          end;
758 8          act$buf$adr = act$buf$adr + bcb$len;
759 8          bcb.link = act$buf$adr;
760 8          bcb$adr = bcb$adr + bcb$len;
761 8          end;
762 7          bcb$adr = bcb$adr - bcb$len;
763 7          bcb.link = 0;
764 7          bcb$adr = bcb$adr + bcb$len;
765 7          end;
766 6          end;
767 5          i = i + 1;
768 5          do while((drives and 1) <> 1) and (i < 16);
769 6          drives = shr(drives,1);
770 6          i = i + 1;
771 6          end;
772 5          drives = shr(drives,1);
773 5          end;
774 4          ii = ii + 1;
775 4          drives = psect(ii).drives;
776 4          end;

777 3          do i = 0 to 16;
778 4          mem$tbl(i).len = sav$mem$len(i);
779 4          end;

780 3          do i = 0 to 15;          /* allocate allocation vectors and */
          /* checksum vectors for banked system */
781 4          dph$adr = drv$tbl(i) + .memory;
782 4          if alloc(i) <> 0 then
783 4          do;
784 5          dph.alv = act$buf$adr;
785 5          act$buf$adr = act$buf$adr + alloc(i);
786 5          end;
787 4          if chk(i) <> 0 then
788 4          do;
789 5          dph.csv = act$buf$adr;
790 5          act$buf$adr = act$buf$adr + chk(i);
791 5          end;
792 4          end;

793 3          end;

794 2          end setbuf;
795 1          end setup$buffers;

```

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

MODULE INFORMATION:

CODE AREA SIZE = 2153H 8531D
VARIABLE AREA SIZE = 00B9H 185D
MAXIMUM STACK SIZE = 0006H 6D
1077 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

0000 GENCPM				
0000 GENCPM				
57E6 MEMORY	5402 FCBIN	5404 BIOSFCB	5428 RESFCB	544C BNKFCB
5470 FCBOUT	5494 DATAFCB	5488 OFFSET	54B9 PRGSIZ	54BB BUFSIZ
54BD CODSIZ	54BF BIOSPG	54C0 SCBPG	54C1 RESPG	54C2 BNKPG
54C3 BNKOFF	54C4 RESLEN	54C5 NONBNK	54C6 DMA	0180 HEXASCII
54C8 LNBFR	57E6 SCTBFR	54D6 FCBMSG	54E3 QUERY	
082E SYSTEMRESET		0837 WRITECONSOLE		54E4 CHAR
084E PRINTCONSOLEBUFFER		54E5 BUFFERADDRESS		
0865 READCONSOLEBUFFER		54E7 BUFFERADDRESS		089C CRLF
08A7 ERROR	54E9 TERMCODE	54EA ERRTYPE	54EB ERRMSGADR	08FC OPENFILE
54ED FCBADDRESS	091E CLOSEFILE	54EF FCBADDRESS	092E DELETEFILE	54F1 FCBADDRESS
093E READRECORD	54F3 FCBADDRESS	095E WRITERECORD		54F5 FCBADDRESS
097E CREATEFILE	54F7 FCBADDRESS	09A7 SETDMAADDRESS		54F9 DMAADDRESS
09B7 READRANDOMRECORD		54FB FCBADDRESS	09D7 WRITERANDOMRECORD	
54FD FCBADDRESS	09F7 SETRANDOMRECORD		54FF FCBADDRESS	5501 AUTOMATIC
5502 DISPLAY	5503 NMBSECT	57E6 LINK	5505 BIOSATTS	550B RESATTS
5511 BNKATTS	5517 RESBIOSLEN	5518 RESBASE	5519 PGDIF	
551A XMOVEIMPLEMENTED		551B OSTOP	551D SYSTEMDATA	551E COMMONLEN
5520 BANKEDLEN	5521 SYSENTRY	561D PRMSGPTR	561E DONTHASH	561F WORDADR
5621 LEN	5622 OFF	5624 RESFLG	5625 SAVEMENTOP	5626 DRVTLADR
5628 DPHADR	562A DPBADR	562C HASHDATA	564C HASHSPACE	564E ALLOC
566E ALLOCSpace	5670 CHK	5690 CHKSPACE	0A07 MOVEF	5692 COUNT
5693 SOURCEADR	5695 DESTADR	0A36 SHIFLEFT	5697 PATTERN	5699 COUNT
0A56 UPPER	569A B	0A88 VALDDRIVE	569B DRV	
0AB4 GETRESPONSE		569C VALADR	0AFB DSPLYHEX	569E VAL
0B24 DSPLYHEXADR		569F VAL	0B45 DSPLYHEXHIGADR	
56A1 VAL	0B5A DSPLYDECAADR		56A2 VAL	56A4 BIG
56A6 DIGIT	56A7 I	56AB PDIGIT	0BE9 DSPLYPARAM	56A9 VAL
56AA BASE	0C1C GETPARAM	56AB STRINGADR	56AD VALADR	56AF PBASE
56B0 BASE	56B1 CHAR	56B2 LBINDX	0D0A PROMPTREAD	0D9B GETSEG
56B3 TYPE	56B4 RECORDSIZE	56B6 K	56B7 SEGNO	56B8 MAXATTR
0F64 PLM	13C7 STASCIHEX	56BA STRINGADR	56BC VAL	56BE I
1443 SETUPSCB	56BF SCBADR	14F9 GETDRVTLADR		56C1 TEMPADR
56C3 TEMP3	1549 PAGECHOP	56C5 I	15D1 GETXMOVE	56C6 XMOVEADR
15F7 DISPLAYLAYOUT		56C8 STRINGADR	56CA BASE	56CC LENGTH
16A9 RELOCModule		56CD FCBADR	16D1 LOAD	56CF FCBADR
56D1 ATTSADR	56D3 I	56D4 RDCNT	179F WRTRUF	56D5 WRTLEN
56D6 WRTOFFADR	56D8 I	56D9 WRTCNT	1848 GETFILEINFO	
56DA FCBADR	56DC ATTSADR	57E6 HEADERRECORD		1899 GETATTS
1913 NEEDTBL	56DE ALLSOME	56DF I	19E9 SETUPHASH	56E0 I
56E1 J	56E2 PRINTED	56E3 SEGNO	56E4 SEGONO	56E5 HBANK
56E6 NOHASH	56E7 SIZE	56E9 HATTR	56EB MAXATTR	56ED MAXOATTR
56EF NOHASH	1C00 GETALLOCCHK		56F0 I	56F1 DBLALLOC
56F2 PRINTED	1DAC SETUPMENSEGTBL		56F3 I	56F4 J
56F5 OK	56F6 ACCEPT	56F7 MLOW	56F8 MHIGH	56F9 TLOW
56FA THIGH	56FB MEMTEMP	218C GETDEFAULTFILE		56FD RET
21E8 SETUPSYSTEMDAT		56FE I	56FF J	5700 OK
5701 TEMP	229E ERR1	23AC SETUPCPM80SYS		5702 I
2451 INITIALIZATION		5703 I		
0000 SETUPBUFFERS				
57E6 MEMORY	2A1A SETBUF	5704 I	5705 J	5706 K
5707 II	5708 SAVE	5709 DATACNT	570A TEMP	570B FIRSTDIR
570C FIRSTDTA	570D FIRSTDRIVE	570E OTHERBANKS	570F OK	5710 PERMMEDIA
5711 PRINTED	5712 LINKCNT	5713 SEGNO	5714 BIRDATAFIELD	
5715 MEMSAVTBL	5737 SAVMEHLEN	5748 RECSIZ	574A DRIVES	574C SAVEDPHADR
574E SAVEBCBADR	5750 BCBcnt	5752 BCBBUFSIZ	5754 BUFSpace	5756 DATAADR
5758 LBUFADR	575A ACTBUFADR	575C LHEADADR	575E MAXDIRBUF	5760 MAXATTR
5762 TPA	5764 DEFINEDDRIVES		5766 BCBLen	5767 BCBBUFCNT
5769 BCBBUFPTR	576A BCBADR	576C PSECT	4262 DISPSpace	57AC SEGO
57AE ORANKS	57B0 II	4354 GETSPACE	57B1 INDEX	57B2 PARM
57B3 COUNT	57B4 II	57B5 I	57B6 J	57B7 K
57B8 SAVE	57B9 SEGNO	57BA RSIZE	4607 DRIVENOTDEFINED	

CPM PLUS V3.0

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. = *CP3-135*

57BC VAL	2FDF ERR4	329C ERR5	340D NOTOK	
0000 GETSYSDEFAULTS				
57E6 MEMORY	57BD SYMBOL	57E6 BUFFER	46C3 GETDEF	
57C5 BUFFERINDEX		57C6 INDEX	57C7 ENDOFFILE	57C8 LINECOUNT
470A ERR	57CA TERMCODE	57CB MSGADR	57CD SAVEDISPLAY	
474D INCPTR	4774 GETCHAR	57CE CHAR	47FA GETSYN	57CF I
57D0 SYNCHAR	57D1 GOTSYS	4911 GETVAL	57D2 FLAGS	57D3 I
57D4 VALCHAR	57D5 VALADR	57D7 BASE	57D8 INC	57D9 LNBFRINDEX
4B88 COMPARESYM	57DA I	57DB J	57DC FOUND	
0000 CREATEDEFAULTS				
57E6 MEMORY	57E6 OBUF	4C0B CRTDEF	57DD FLAGS	57DE SYMBOLDONE
57DF I	57E0 J	57E1 OBUFINDEX	57E2 INC	57E3 VALADR
4E27 INCOBUFINDEX		4E62 EMITASCIHEX		57E5 DIG
0000 RELFIX				
4F8C BITMAP	500C BTMPTP	4F74 DEFAULT	4F7A ENDT	4F86 ERRTN
4F90 EXITRE	4EC8 NOFILL	4EDC PGREL0	4F06 PGREL2	4F0A PGREL3
4F7C PGREL4	4EA3 RELFIX	4F1C TEST2	4F2E TEST3	4F3B TEST4
4F42 TEST5	4F4C TEST6	4F67 TEST7		
0000 DATMOD				
5133 BCKSPC	5130 BDRIVE	512D BNKSWT	512F BNKTOP	5131 CONPAG
5132 CONWID	523B CRDATF	523C DBLALV	5136 HASH	512B LERROR
5146 HEMTBL	512E HEMTOP	5135 NUMSEG	512C PRTHSG	523D QUEST
519B RECORD	5134 RUBOUT	500D SYMTBL		

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

0000 GENCPM#

0000 GENCPM#

082E 53	082E 54	0836 55	0837 56	083B 58
0842 59	084D 60	084E 61	0854 63	085B 64
0864 65	0865 66	086B 69	0871 70	0878 72
087F 73	0880 74	0880 75	0889 76	089B 77
089C 78	089C 79	08A1 80	08A6 81	08A7 82
08B4 85	08B9 86	08BF 87	08C7 88	08CF 89
08D5 90	08D8 91	08DF 92	08E2 93	08EE 95
08F8 96	08FB 97	08FB 98	08FC 99	0902 102

0914 103	091E 104	091E 105	0924 107	092D 108
092E 109	0934 111	093D 112	093E 113	0944 115
0952 117	095D 118	095D 119	095E 120	0964 122
0972 124	097D 125	097D 126	097E 127	0984 130
0992 132	099D 133	099D 134	09A6 135	09A7 136
09AD 138	09B6 139	09B7 140	09BD 142	09CB 144
09D6 145	09D6 146	09D7 147	09DD 149	09EB 151
09F6 152	09F6 153	09F7 154	09FD 156	0A06 157
0A07 211	0A16 214	0A1E 215	0A1F 216	0A35 217
0A36 218	0A3E 221	0A46 222	0A4A 223	0A55 224
0A56 225	0A5A 227	0A62 228	0A65 229	0A7C 230
0A84 231	0A88 232	0A88 233	0A8C 235	0AA3 236
0AA6 237	0AB1 238	0AB4 239	0AB4 240	0ABA 243
0ABF 244	0AC8 245	0AD0 246	0AD5 247	0ADB 248
0AE1 249	0AE9 250	0AEA 251	0AFA 252	0AFB 253
0AFF 255	0B13 256	0B23 257	0B24 258	0B2A 260
0B2F 261	0B37 262	0B3F 263	0B44 264	0B45 265
0B49 267	0B59 268	0B5A 269	0B60 274	0B65 275
0B6A 276	0B70 277	0B7C 278	0B86 280	0B94 281
0BA0 282	0BA5 283	0BAC 284	0BBA 285	0BBD 286
0BC4 288	0BCB 289	0BD0 290	0BD0 291	0BDE 292
0BE8 293	0BE8 294	0BE9 295	0BEF 297	0BF4 298
0BFC 300	0C01 301	0C0B 302	0C0E 304	0C15 305
0C15 306	0C1B 307	0C1C 308	0D0A 316	0D0A 317
0D12 318	0D1B 320	0D20 321	0D27 322	0D35 323
0D3C 324	0D50 326	0D55 327	0D58 329	0D5D 330
0D64 331	0D64 332	0D6E 333	0D76 334	0D7C 335
0D7F 337	0D8A 338	0D8A 339	0D8F 340	0D94 341
0D9A 342	0C2B 343	0C2E 344	0C36 346	0C39 347
0C3A 348	0C3A 349	0C3F 350	0C59 351	0C61 353
0C68 354	0C6D 355	0C72 356	0C75 358	0C7D 360
0C82 361	0C85 363	0C8D 364	0CA4 366	0CAD 367
0CB8 368	0CBD 369	0CBD 370	0CC7 372	0CE3 373
0CE6 375	0CF0 376	0CFB 377	0CFE 378	0D03 379
0D03 380	0D03 381	0D03 382	0D06 383	0D09 384
0D9B 385	0DA3 388	0DAB 389	0DAE 390	0DB3 391
0DBA 392	0DC9 393	0DE6 394	0DEE 396	0E2A 398
0E30 399	0E4A 400	0E4A 401	0E4D 403	0E89 405
0EBF 406	0EA9 407	0EA9 408	0EA9 409	0EB3 410
0ECB 411	0EDA 412	0F36 414	0F3C 415	0F56 416
0F56 417	0F60 418	0F64 419	0F64 420	13C7 421
13D1 426	13D6 427	13DC 428	13F6 429	140E 430
142A 431	1442 432	1443 433	1443 436	145F 437
146A 438	1475 439	1480 440	148B 441	1494 442
14A5 443	14AD 444	14BC 445	14CC 446	14DE 447
14ED 448	14F8 449	14F9 450	14F9 454	14FF 455
150D 456	1513 457	1520 458	152B 459	152D 460
153A 461	1544 462	1548 463	1549 464	1549 466
154F 467	1554 468	1561 470	156F 471	1583 473
1597 474	15A8 475	15AD 476	15AD 477	1587 478
15C3 479	15CB 480	15D0 481	15D0 482	15D1 483
15D1 486	15DD 487	15E7 488	15F0 489	15F3 490
15F6 491	15F7 492	1606 496	160C 497	1614 498

COPYRIGHT ©1982

DIGITAL RESEARCH

P.O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

1619 499	1621 500	1626 501	162D 502	1634 504
1649 505	1651 506	1663 507	166B 508	1685 509
168D 510	16A0 511	16A8 512	16A8 513	16A9 514
16AF 516	16B5 517	16BD 519	16C8 520	16C8 521
16D0 522	16D1 523	16DB 528	16E5 529	16F1 530
16FF 531	170E 532	1720 533	1735 535	1740 536
1740 537	174E 538	1753 539	175D 540	1774 541
177C 542	1783 543	1786 544	1794 545	179E 546
179F 547	17A7 551	17AF 553	17B8 554	17C7 555
17CF 556	17D7 557	17DD 558	17F9 559	1805 560
1812 561	1824 562	182A 563	1834 564	183A 565
1847 566	1847 567	1848 568	1899 573	1899 574
18A8 575	18BA 576	18C7 577	18D2 578	18D8 579
18E0 580	18EC 581	18FA 582	190A 583	1912 584
1848 585	1850 587	185C 588	1868 589	186B 591
1871 592	1877 593	187A 594	187A 595	1880 596
1886 597	1889 598	188F 599	1895 600	1898 601
1913 602	1913 604	1918 605	1925 606	1928 607
1936 608	194A 610	195E 611	196D 612	19D6 613
19DB 614	19DB 615	19E5 616	19E9 617	19E9 618
19E9 622	19EE 623	19F3 624	1A01 625	1A15 626
1A29 628	1A3A 629	1A3F 630	1A50 632	1A58 634
1A5D 635	1A63 636	1A63 637	1A73 638	1A83 639
1A95 640	1A9B 641	1AA4 642	1AAA 643	1AB8 644
1ABB 645	1ACA 647	1AD7 648	1ADA 649	1AE2 651
1AE7 652	1AFA 653	1B05 654	1B08 656	1B19 657
1B27 659	1B45 660	1BA2 661	1BD4 662	1BD4 663
1BD4 664	1BD4 665	1BD4 666	1BDE 667	1BEA 669
1BEF 670	1BF7 671	1BFF 672	1BFF 673	1C00 674
1C00 677	1C0E 678	1C1E 679	1C2E 680	1C38 681
1C48 682	1C50 683	1C55 684	1C5B 685	1C61 686
1C66 687	1C74 688	1C88 690	1C9C 691	1CAC 692
1CBD 694	1CC3 695	1CCC 696	1CD1 697	1CD6 698
1CFE 699	1D13 700	1D13 701	1D24 703	1D2A 704
1D33 705	1D38 706	1D3D 707	1D5D 708	1D6F 709
1D97 710	1D97 711	1D97 712	1DA1 713	1DA8 714
1DAB 715	1DAC 716	1DAC 719	1DB2 720	1DBC 721
1DC2 722	1DC7 723	1DCC 724	1DD2 725	1DD8 726
1DE4 727	1DEA 728	1DF1 729	1DF6 730	1DFD 731
1E02 732	1E09 733	1E0F 734	1E17 735	1E1D 736
1E2C 737	1E31 738	1E39 739	1E4A 740	1E64 741
1E99 742	1EB2 744	1EBD 745	1EC0 746	1ED9 748
1EE4 749	1EE7 751	1EFA 752	1F1B 753	1F36 755
1F3C 756	1F5A 757	1F83 758	1F86 760	1F8D 761
1F92 762	1F97 763	1FA5 764	1FB8 765	1FD2 766
1FFF 768	2018 770	2023 771	2028 772	202B 773
2042 775	2048 776	205B 777	2060 778	2063 779
207A 781	2080 782	209E 783	20C7 784	20CC 785
20CC 786	20CC 787	20D3 788	20D6 789	20D6 790
20D6 791	20D9 792	20E3 793	20E6 794	20F5 795
20FD 796	2106 798	210C 799	2114 800	2114 801
2127 802	213E 803	2145 805	214B 806	2162 807
2162 808	2165 809	216F 810	2174 811	2179 812
217F 813	2185 814	2188 815	218B 816	218C 817
218C 819	2192 820	21A0 822	21AC 823	21B8 824
21BE 825	21C3 826	21C9 827	21CC 828	21D3 829
21D6 830	21DC 831	21DF 833	21E4 834	21E7 835
21E7 836	21E8 837	21E8 839	21ED 840	21F0 841
21F8 842	21FE 843	2201 844	2207 845	220D 846
2213 847	2216 848	2219 849	221F 850	2225 851
2228 852	2228 853	2231 854	2238 855	2244 856
224B 857	2251 858	225B 859	2264 860	226B 861
2271 862	2277 863	227D 864	2280 865	2286 866
228C 867	2292 868	2295 869	2298 870	229E 871
22A4 872	22AD 873	22B3 874	22B9 875	22C1 877
22CD 878	22D9 879	22DC 880	22E2 881	22E2 882

COPYRIGHT ©1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

22E5 883	22E8 884	22EE 885	22FA 886	2308 887
2311 888	2317 889	231D 890	2320 891	2327 892
232E 894	2334 895	2340 896	2343 897	2349 898
234F 899	2355 900	2358 901	235B 903	2361 904
2364 905	236A 906	2370 907	2373 908	2378 909
2378 910	237D 911	2382 912	2385 913	238B 914
2391 915	2394 916	2397 917	239D 918	23A4 919
23AB 920	23AC 921	23AC 923	23B2 924	23B8 925
23BE 926	23C3 927	23D1 928	23DC 929	23E6 930
23F4 931	23FF 932	2409 933	240E 934	2421 935
2426 936	2432 937	243E 938	2444 939	244A 940
2450 941	2451 942	2451 944	2463 945	2471 946
2481 947	248B 948	2491 949	2499 951	249E 952
24A1 953	24AF 954	24BA 955	24C4 956	24C7 958
24CC 959	24CF 960	24CF 961	24D7 963	24DC 964
24DC 965	24E1 966	0F64 967	0F69 968	0F6E 969
0F71 970	0F74 971	0F77 972	0F7A 973	0F86 974
0F94 975	0F9F 976	0FA9 977	0FBD 978	0FCD 979
0FD6 980	0FD9 981	0FE1 983	0FEF 984	1003 985
1006 987	100D 988	101D 989	101D 990	1024 992
1027 993	102E 995	104D 996	105D 997	1063 998
1066 999	106E 1000	107C 1001	108E 1002	1098 1003
1098 1004	10A0 1005	10A8 1006	10AB 1007	10AE 1008
10AE 1009	10B4 1010	10BA 1011	10C0 1012	10C7 1013
10E2 1014	10FA 1015	1102 1017	1108 1018	110E 1019
1118 1020	111E 1021	1123 1022	1126 1024	1135 1025
114A 1026	1162 1027	116C 1028	1172 1029	1178 1030
117E 1031	118B 1032	1195 1033	119B 1034	119B 1035
11AA 1036	1180 1037	1189 1038	11BF 1039	11C2 1040
11D8 1041	11E7 1042	11EF 1043	120A 1044	1222 1045
122C 1046	1234 1047	123B 1049	1240 1050	1246 1051
124F 1052	1255 1053	1260 1054	1278 1055	1290 1056
129A 1057	12A2 1058	12A2 1059	12AA 1060	12B2 1061
12C5 1062	12CF 1063	12EB 1064	1307 1065	130F 1066
131D 1067	1329 1068	1335 1069	133A 1070	133D 1071
1340 1072	1346 1073	134C 1074	1358 1075	1364 1076
136A 1077	1370 1078	1375 1079	137B 1080	1381 1081
1387 1082	138E 1084	1393 1085	1399 1086	13A0 1087
13AC 1088	1388 1089	13BB 1090	13BB 1091	13C0 1092
13C6 1093	13C7 1094			

0000 SETUPBUFFERS#

2A1A 94	4262 112	4262 115		
426B 116	427A 117	4293 118	42B6 119	42D6 120
42DD 121	42E3 122	42E9 123	42F6 124	42FD 126
4303 127	430B 128	4312 130	4318 131	4320 132
4320 133	4340 134	434B 135	434D 136	434D 137
4350 138	4353 139	4354 140	435A 145	436D 146
4374 148	4388 149	438B 151	4390 152	4398 154
43AC 155	4388 156	43C5 158	43CA 159	43D1 160
43D6 161	43DB 162	43EA 163	4405 164	4424 166
442A 167	4430 168	4430 169	4437 170	4456 171
4459 172	4463 173	4474 175	447F 176	4482 177
4485 178	44A4 179	44AB 180	44AE 182	44C2 183
44D1 184	44DB 186	44E0 187	44E7 188	44EC 189
44F1 190	4500 191	451B 192	453A 194	4540 195
4546 196	4546 197	454D 198	456C 199	456F 200
4579 201	458E 203	459C 204	45AB 206	45B6 207
45B9 208	45B9 209	45BC 211	45CD 213	45DB 214
45DB 215	45DE 216	45FD 217	45FD 218	4604 219
4604 220	4604 221	4607 222	4607 223	460B 225
4624 226	4627 227	4632 228	4635 229	2A1A 230
2A2B 231	2A3C 233	2A50 234	2A77 236	2A87 237
2AA6 238	2AB8 239	2AB8 240	2AC9 242	2AD4 243
2AD4 244	2AD4 245	2ADB 246	2AE9 247	2AEF 248
2AFC 249	2B23 250	2B36 251	2B3D 252	2B51 253
2B63 254	2B71 255	2B99 257	2BBC 258	2BCE 259

COPYRIGHT ©1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

2BCE 260	2BD5 261	2BDC 262	2BEA 263	2C10 264
2C30 265	2C37 266	2C3E 267	2C46 268	2C4B 269
2C50 270	2C57 271	2C5E 272	2C64 273	2C67 274
2C6B 275	2C70 276	2C84 277	2C8B 278	2C9E 279
2CAC 280	2CD2 281	2CE4 282	2CEB 283	2CF0 284
2CF8 285	2CFD 286	2D11 287	2D33 288	2D3A 290
2D40 291	2D45 292	2D45 293	2D4B 294	2D5C 295
2D62 296	2D67 297	2D86 298	2D93 299	2D97 300
2D9A 301	2DA7 302	2DAF 303	2DC3 304	2DD0 305
2DE1 306	2DF2 307	2DFA 308	2DFF 309	2E10 311
2E13 312	2E27 313	2E2E 315	2E35 317	2E3A 318
2E4C 319	2E52 320	2E5B 321	2E61 322	2E64 324
2E74 325	2E7A 326	2E83 327	2E89 328	2E9E 329
2EA1 330	2EB6 332	2EC8 333	2ECB 334	2EDD 335
2EDD 336	2EE0 338	2EF0 339	2EF6 340	2EFF 341
2F1A 342	2F21 344	2F26 345	2F3C 346	2F47 347
2F4D 348	2F56 349	2F71 350	2F74 351	2F74 352
2F74 353	2F8E 355	2F94 356	2F9C 357	2FA2 358
2FB9 359	2FB9 360	2FCF 362	2FDF 363	2FE5 364
2FFC 365	3002 366	3008 367	3010 369	301C 370
3027 371	302A 372	3035 373	3038 374	304C 375
304C 376	304F 377	3052 378	3060 379	3063 380
3063 381	3074 383	3079 384	308E 386	309B 387
30AD 388	30B0 390	30B3 391	30C7 392	30CE 394
30D6 396	30DD 398	30E2 399	30E7 400	30F9 401
30FF 402	3108 403	310E 404	3111 406	311B 407
311D 408	311D 409	311D 410	3122 411	312A 413
313A 414	3140 415	3149 416	314F 417	3164 418
3167 419	317C 420	3191 422	31A3 423	31A8 424
31AB 425	31A8 426	31AB 428	31BB 429	31C1 430
31CA 431	31E5 432	31F9 433	3200 435	3205 436
321B 437	3226 438	322C 439	3235 440	3250 441
3253 442	3253 443	3253 444	3269 446	3270 448
3275 449	3289 450	328C 452	329C 453	32A2 454
3289 455	32BF 456	32C5 457	32CD 459	32D9 460
32E4 461	32E7 462	32F2 463	32F5 464	3309 465
3309 466	330C 467	330C 468	3321 469	3324 471
332F 473	333F 474	3345 475	335A 476	335D 477
335D 478	336B 479	336E 480	336E 481	336E 482
336E 483	3373 484	338B 485	338C 486	33AB 487
338B 488	338C 489	33BF 490	33CC 491	33CF 492
33D3 493	33E7 494	33EA 495	33EF 496	33FA 497
33FB 499	33FE 500	3404 501	340A 502	340D 503
340D 504	3410 505	3420 506	342C 507	3442 509
346E 511	3475 512	347C 513	34A1 514	34BB 515
34BB 516	34E7 518	34EE 520	34F2 521	3507 522
352C 523	352F 524	3554 525	356E 526	356E 527
356E 528	3575 529	358D 530	3594 532	359E 533
35D1 534	35D9 535	35E0 536	35E7 537	35EE 538
35F5 539	35FD 540	3610 541	3619 542	361F 543
3630 544	3639 545	3640 546	3643 547	3648 548
365C 549	367E 550	3683 551	36A2 552	36AF 553
36B3 554	36B6 555	36C3 556	36CB 557	36DF 558
36F7 560	370D 562	3712 563	3722 564	3739 565
3749 566	3757 567	3765 568	3768 570	376E 571
3790 572	379B 573	37A1 574	37AB 575	37AB 576
37AB 577	37C3 579	37D9 581	37DE 582	37EE 583
3805 584	3815 585	3823 586	3831 587	3834 589
383A 590	3860 591	3871 592	3882 593	3890 594
3896 595	38A3 596	38A3 597	38A3 598	38A7 599
38C6 600	38D3 601	38D7 602	38DA 603	38E7 604
38EA 605	38EE 606	3902 607	3905 608	3913 609
3926 611	393A 612	3947 613	395C 614	395C 615
3963 616	3971 617	3985 618	3998 620	39AB 621
39BD 622	39BD 623	39D0 625	39E0 626	39F5 627
39F5 628	39FC 629	39FF 631	3A0D 632	3A33 633

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

3A3A 634	3A4B 635	3A6A 636	3A87 637	3AA1 638
3AAB 639	3AB2 640	3AB9 641	3AC0 642	3ADA 643
3AF9 645	3B09 646	3B1F 647	3B2C 648	3B3F 649
3B42 651	3B57 652	3B61 653	3B74 654	3B87 655
3B87 656	3B92 657	3BA2 658	3BB0 659	3BB7 660
3BC3 661	3BC9 662	3BDA 663	3BE3 664	3BEA 665
3BED 666	3BF2 667	3BF7 668	3C06 669	3C21 670
3C40 672	3C46 673	3C4C 674	3C4C 675	3C53 676
3C59 677	3C63 678	3C6B 680	3C8A 682	3CB6 683
3CC6 684	3CF5 685	3CF5 686	3CF5 687	3CFA 688
3D0E 689	3D30 690	3D35 691	3D54 692	3D61 693
3D65 694	3D68 695	3D75 696	3D7D 697	3D91 698
3DA9 700	3DBF 702	3DC5 703	3DE7 704	3DF2 705
3DF8 706	3E02 707	3E05 709	3E11 710	3E21 711
3E29 712	3E31 713	3E4E 714	3E65 715	3E6A 716
3EAB 717	3EB4 718	3EDD 719	3EEB 720	3EF5 721
3F03 722	3F0A 723	3F18 724	3F22 725	3F30 726
3F30 727	3F30 728	3F48 730	3F5E 732	3F64 733
3F86 734	3F91 735	3F97 736	3FA1 737	3FA4 739
3FB0 740	3FC0 741	3FCB 742	3FD0 743	3FED 744
4002 746	4019 747	401E 748	405F 749	407D 750
40A6 751	40A9 753	40AE 754	40BE 755	40D5 756
40DE 757	40DE 758	40EC 759	40F6 760	4104 761
410B 762	4119 763	4123 764	4131 765	4131 766
4131 767	4135 768	4154 769	4161 770	4165 771
4168 772	4175 773	4178 774	417C 775	4190 776
4193 777	41A1 778	41C1 779	41C8 780	41D6 781
41EA 782	41FD 784	420D 785	4222 786	4222 787
4235 789	4245 790	425A 791	425A 792	4261 793
4261 794				

0000 GETSYSDEFAULTS#

46C3 47	470A 52	4712 55	4718 56	
471D 57	4723 58	472B 59	4731 60	4739 61
4740 62	4743 63	4746 64	474C 65	474D 66
474D 67	4755 69	475A 70	4767 71	476C 72
476F 73	4773 74	4774 75	4774 77	4777 78
4784 79	47A8 80	47B0 81	47B7 82	47BA 83
47C7 84	47CA 85	47E1 86	47E9 87	47F1 88
47F6 89	47FA 90	47FA 91	47FA 94	47FF 95
480F 96	481D 97	482B 98	482F 99	4835 100

483A 101	4864 102	4871 103	4877 104	487B 105
487E 106	489E 107	48A4 108	48A7 109	48AE 111
48C5 112	48CD 114	48D5 115	48E0 116	48E8 117
48F0 118	4904 119	490A 120	490D 121	490D 122
490D 123	4910 124	4911 125	4911 130	4917 131
491C 132	493C 133	4949 134	494D 135	4952 136
4958 137	495B 138	496F 139	4975 140	4978 141
497D 142	4980 143	4989 145	49A4 146	49B1 147
49B8 149	49C2 150	49CD 151	49E4 152	49EF 153
49F9 154	4A1F 155	4A1F 156	4A2E 158	4A4E 159
4A53 160	4A57 161	4A5B 162	4A5B 163	4A64 165
4A6E 167	4A89 168	4A9C 169	4AA4 170	4AA7 171
4AB1 173	4AC4 174	4AC7 176	4ACC 177	4AD1 178
4AE1 179	4AF3 180	4AF8 182	4AFF 183	4B01 184
4B06 185	4B09 187	4B11 188	4B19 190	4B21 191
4B38 193	4B41 194	4B4C 195	4B51 196	4B51 197
4B5B 198	4B78 199	4B80 200	4B80 201	4B80 202
4B87 203	4B87 204	4B87 205	4B87 206	4B88 207
4B88 210	4B8D 211	4B92 212	4BA4 213	4BA9 214
4BDE 215	4BE2 216	4BE5 217	4BED 218	4BF5 219
4BF9 220	4BFC 221	4C03 222	4C06 223	4C0A 224
46C3 225	46C9 226	46CF 227	46D4 228	46D9 229
46E0 230	46E3 231	46EA 233	46F0 234	46F8 235
46FE 236	4706 237	4706 238	4709 239	

0000 CREATEDEFAULTS#

4C0B 28

COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

4E27 32	4E27 33	4E2F 35	4E35 36	4E43 37
4E4E 38	4E55 39	4E5A 40	4E5D 41	4E61 42
4E62 43	4E66 45	4E69 46	4E86 47	4E89 48
4EA2 49	4C08 50	4C11 51	4C17 52	4C1D 53
4C22 54	4C30 55	4C35 56	4C4B 57	4C50 58
4C62 59	4C70 60	4C73 61	4C97 62	4C9E 63
4CA8 64	4CB0 66	4CBA 67	4CCC 69	4CD4 71
4CE3 72	4CE6 74	4CF7 75	4CF7 76	4CF7 77
4CF7 78	4CFA 79	4D05 80	4D08 81	4D13 82
4D16 83	4D21 84	4D4D 85	4D57 87	4D5A 88
4D6A 89	4D6D 91	4D77 93	4D7A 94	4D82 95
4D90 96	4D9B 97	4D9E 99	4DA5 100	4DAF 102
4DB2 103	4DBD 104	4DC4 105	4DCB 106	4DCE 107
4DD9 108	4DE0 109	4DE7 110	4DE7 111	4DE7 112
4DE7 113	4DEA 114	4DF5 115	4DF8 116	4E03 117
4E07 118	4E0A 119	4E11 120	4E1A 121	4E20 122
4E26 123				

0000 RELFIX#

0000 DATMOD#

COPYRIGHT ©1982
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____